LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-SR-677652

Rutgers University Subcontract B611610 Final Report

T. Eliassi-Rad

September 30, 2015

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Department of Computer Science
School of Arts and Sciences
Rutgers, The State University of New Jersey
110 Frelinghuysen Road
Piscataway, NJ 08854-8019

http://www.cs.rutgers.edu/~eliassi
eliassi@cs.rutgers.edu

p. 848-445-8851
f. 732-445-0537

September 3, 2015

Mr. Brian Gallagher
7000 East Avenue
Livermore, CA 94550-9234

Dear Mr. Gallagher:

This is the final report for LLNL Subcontract B611610 with Rutgers University. Since January 2015, we have investigated incomplete networks and methods for enriching them. Networked representations of physical and social phenomena are often incomplete because the phenomena are partially observed. Working with incomplete networks can skew analyses; and hoping to acquire the full data is often unrealistic. However, one may be able to collect data selectively to enrich the incomplete network.

We call the aforementioned task of enriching incomplete networks *active graph probing*; and have studied two specific cases:

1. *Active node probing*: Given an incomplete (i.e., partially-observed) network, which nodes should we actively probe in order to achieve the highest accuracy for a given network feature? For example, consider a cyber-network administrator who observes only a portion of the network at time t and wants to accurately identify the most important (e.g., highest PageRank) nodes in the complete network. She has a limited budget for probing the network. Of all the nodes she has observed, which should she probe in order to most accurately identify the important nodes? We propose a novel and scalable algorithm, MaxOutProbe, and evaluate it w.r.t. four network features (largest connected component, PageRank, core-periphery, and community detection), five network sampling strategies, and seven network datasets from different domains. Across a range of conditions, MaxOutProbe demonstrates consistently high performance relative to several baseline strategies.
2. *Active edge probing*: Suppose that one is given a sample of a larger graph and a budget to learn additional neighbors of nodes within the sample, with the goal of obtaining the most valuable information about the graph as a whole. Which nodes should be further explored? For example, a network administrator may have been given an incomplete map of her network, and can choose machines in which to run traceroutes, thus obtaining additional edges. Which machines should she select? We introduce ε-WGX, a multi-armed bandit-based algorithm for identifying which nodes in a graph sample should be probed. Our experiments compare ε-WGX to several baseline-probing algorithms on four real network datasets using samples generated by four popular sampling methods. We consider two reward functions: (1) bringing in new nodes into the sample and (2) closing triangles within the sample, and show that ε-WGX significantly improves over random probing. For example, averaged over all sample types, at the task of closing triangles within the sample, ε-WGX improves over random probing by 29%.

Attached please find two reports that detail our finding on active node probing and active edge probing.

Sincerely,

Tina Eliassi-Rad
Associate Professor
Dept. of Computer Science
Rutgers University

Cc: Gary Ward
Encl: Two reports

Probing Strategies for Incomplete Networks

Sucheta Soundarajan* Tina Eliassi-Rad* Brian Gallagher† Ali Pinar‡
*Rutgers University †Lawrence Livermore National Laboratory ‡Sandia National Laboratories
*{s.soundarajan, eliassi}@cs.rutgers.edu †bgallagher@llnl.gov ‡apinar@sandia.gov

ABSTRACT

Given an incomplete (i.e., partially-observed) network, which nodes should we *actively probe* in order to achieve the highest accuracy for a given network feature? For example, consider a cyber-network administrator who observes only a portion of the network at time t and wants to accurately identify the most important (e.g., highest PageRank) nodes in the complete network. She has a limited budget for probing the network. Of all the nodes she has observed, which should she probe in order to most accurately identify the important nodes? We propose a novel and scalable algorithm, *MaxOutProbe*, and evaluate it w.r.t. four network features (largest connected component, PageRank, core-periphery, and community detection), five network sampling strategies, and seven network datasets from different domains. Across a range of conditions, *MaxOutProbe* demonstrates consistently high performance relative to several baseline strategies.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and Networks; H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Design, Performance, Experimentation.

Keywords

Networks, graph mining, active learning

1. INTRODUCTION

Suppose that one has an incomplete portion G_{samp} of some larger complete network G_{orig}.[1] To learn more about the structure of G_{orig}, one can probe nodes from G_{samp}. The *active graph probing* problem asks the following question: Which nodes from G_{samp} should be probed to reveal the most useful information (w.r.t. some network feature) about the structure of G_{orig}?

Our work is motivated by problems in cybersecurity and other domains, where one only has a partial observation of the complete network; but for situational awareness purposes, needs the most accurate and informative picture of the complete network. For example, suppose a network administrator has partially observed a network through traceroutes. Which parts of the sampled network should be more closely examined to give the best (i.e., most complete) view of the entire network? With a limited query budget, how should this further exploration be done? Alternatively, suppose that one has obtained a sample of the Twitter network from another researcher. The sample was collected for some other purpose, and so may not contain the most useful structural information for one's purposes. How should one best supplement this sampled data?

This problem is related to previous works on graph sampling and crawling. However, unlike much of the work in graph sampling, we are not attempting to generate a sample from scratch. Instead, we are studying how one can enhance or improve an existing sample, without control over how that sample was generated.

We present MaxOutProbe, a novel algorithm for the active graph probing problem. MaxOutProbe is based on the intuition that successful probes of G_{samp} give us broader knowledge of G_{orig}. That is, they give us information about previously-unseen nodes. MaxOutProbe contains two major steps. First, for each node u in G_{samp}, MaxOutProbe estimates u's true degree in G_{orig}; it also estimates G_{orig}'s transitivity. Second, MaxOutProbe uses these two quantities to estimate the number of nodes outside G_{samp} to which u is adjacent. Nodes that are predicted to have the most neighbors outside the current sample are selected for probing. This probing strategy is appropriate for many network features detailed in the next paragraph.

We consider four network features: the nodes in the largest connected component of the network (LCC), the highest PageRank nodes (PR), community structure (Comms), and the core and periphery of the network (Core-Periphery). We consider samples generated by five popular sampling techniques applied to seven real network datasets from multiple domains, and demonstrate that MaxOutProbe selects nodes for probing

[1]Throughout this paper, when we use the term *complete*, we are referring to the completeness of the data—i.e., no information is missing—rather than to a clique structure.

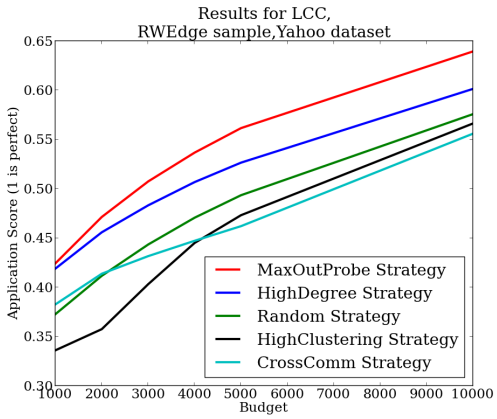

Figure 1: Average results for the LCC feature on Yahoo! IM network for samples collected via Random Walks. The x-axis is the budget on the number of nodes probed. The y-axis is what fraction of the LCC was recovered. MaxOutProbe **is the best strategy at all budgets.**

that are significantly more valuable than those selected by the baseline methods, with respect to the aforementioned four network features. For example, Figure 1 depicts the performance of MaxOutProbe on the LCC feature for a Yahoo! IM network, as compared to several baseline probing strategies.

The **contributions** of our paper are as follows:

- We introduce *active graph probing*, the problem of determining which k nodes in an incomplete (i.e., partially observed) network to probe in order to obtain a network structure that is more accurate with respect to a specific network feature such as the size of the LCC.
- We present MaxOutProbe, a two-step algorithm for selecting which nodes from an incomplete network one should probe.
- We perform an extensive set of experiments spanning seven network datasets, four network features, and five sampling methods (which generate the initial incomplete graph), and compare MaxOutProbe to a variety of baseline probing strategies.
- We show that with respect to the four considered graph features, the graphs obtained by probing nodes according to MaxOutProbe are significantly better than those obtained by probing according to the baseline strategies.

2. PROBLEM STATEMENT: ACTIVE GRAPH PROBING

Formally stated, the *active graph probing* problem is as follows: Given an incomplete, partially observed graph G_{samp} that is a sample graph of a larger, fully observed graph G_{orig}, predict network features (e.g., PageRank) on G_{orig} by computing them on G_{samp}. To aid in this task, we are allowed to probe a specified number b additional nodes in G_{samp} and gain more information about the probed nodes. Let G'_{samp} represent the augmented graph—i.e., G_{samp} with the information obtained from the probes. Our goal is to select b nodes cleverly to maximize the accuracy of our prediction of the network features on G_{orig} using G'_{samp}.

We define probing a node as learning all of its neighbors (e.g., querying Facebook for a list of all friends of a user or learning all e-mail contacts of an individual). We perform these probes in batch,[2] and we only probe nodes that already exist in G_{samp} (that is, there is no 'master list' of nodes in G_{orig} that allows us to probe nodes that we have not yet seen).

The general problem is too broad for a single paper, thus we focus on a special case of the problem, as described below.

1. We know the process by which G_{samp} was sampled from G_{orig} (see Section 3.1). When, as part of the sampling procedure, sampled nodes are fully explored (i.e., all of their neighbors are learned), we know which nodes were sampled. This assumption applies to only two out of our five sampling methods, as the remaining three sampling methods sample edges rather than nodes.[3]
2. We know the fraction of nodes and edges from G_{orig} that are present in G_{samp}. For example, G_{samp} may contain 10% of the edges from G_{orig} and 5% of the nodes from G_{orig}.
3. G_{orig} exhibits a high clustering coefficient, as is typical in real graphs.

Our assumption that we know the process by which G_{samp} was generated is realistic when we have non-adversarial contact with the generator of the dataset, who can provide this information. Our assumption that we know the size of G_{samp} relative to G_{orig} is equivalent to knowing the number of nodes and edges in G_{orig}, which is also realistic. Our third assumption is reasonable for a typical real graph (see, e.g., [27]). However, there are a number of variations on this problem statement: e.g., one might only learn one edge from each probe. We leave such problems for future work.

Figure 2 illustrates the probing process. Red nodes have already been fully explored during sampling (for some sampling methods, as discussed in Section 3, there are no such nodes). Yellow nodes are present in the sample, but have not yet been fully explored: these are the candidates for probing. Green nodes are not yet

[2]There is one exception to this, discussed in Section 4.2.3. In this case, we conduct probes in two sets.

[3]If we eliminate this assumption, then performance of all probing strategies decreases, because we may use much of our budget probing nodes that are already fully explored.

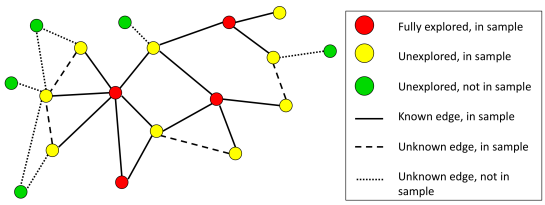

Figure 2: Overview of probing. Red nodes are already fully explored, and we select probes from among the yellow nodes. The goal is to select a yellow node that is adjacent to many green nodes.

present in the sample, and we have no knowledge of them. By probing yellow nodes, we learn about the existence of the green node. We refer to the red nodes as 'fully explored', and the yellow nodes as 'unexplored.'

3. PRELIMINARIES

We consider five sampling methods and four network features.

3.1 Sampling Methods

We consider five sampling methods, each corresponding to real application scenarios. For each method, we sample 10% of the edges from the original graph.

RandNode: We sample nodes at random. The sample contains those nodes plus all of their neighbors. We assume that a list of the selected nodes is available. We select one node at a time until our sample reaches 10% of the total number of edges in G_{orig}.

RandEdge: Random edges are selected uniformly at random. Because edges, rather than nodes, are sampled, no list of sampled nodes is available. The Twitter firehose, for example, provides a 10% sample of tweets (where each retweet represents an edge).

BFS: A breadth-first search is conducted from a random seed node. All nodes except the leaves of the BFS are fully explored, and we assume that a list of the sampled nodes is available. BFS is a common way of crawling the web (see, e.g., [6]).

Random Walk (RW) and **Random Walk w/ Jump (RWJ)**: These sampling methods begin at a random node and repeatedly transition to a random neighbor. In the RWJ method, at each step there is a chance (say 15%) of jumping to a random node. A single edge at a time is observed. Thus, no list of sampled nodes is available. These are common sampling methods for large networks [14].

3.2 Network Features

We have selected four network features, each requiring different levels of knowledge about the network. Knowledge required may be *broad* (i.e., information from many different regions of the network) or *narrow* (i.e., information only about central portions of the network); and *deep* (i.e., thorough exploration of the observed regions) or *shallow* (i.e., cursory information is sufficient).

LCC: We find the largest connected components (LCCs) of G_{orig} and G'_{samp}, and calculate the Jaccard similarity between the two sets of nodes. This feature needs only narrow and shallow knowledge of the network (an un-detailed picture of part of the network).

PageRank: We calculate the Jaccard similarity between the top-1000 PageRank nodes in G_{orig} and the top-1000 PageRank nodes in G'_{samp}. This feature needs a narrow, but deep, view of the network (detailed information about part of the network).

Core-Periphery: For $k \in 2, 4$, we identify the k-cores of G_{orig} and G'_{samp}.[4] We calculate the Jaccard similarities between the detected cores and fringes, and report the average of these two values. The Core-Periphery feature requires a broad, but shallow, view of the network. To receive a good score, G'_{samp} must contain nodes from both central and fringe portions of the network. However, this knowledge need not be very deep; that is, detailed knowledge about clustering and so on is not necessary.

Comms: We find communities in G_{orig} and G_{samp} using the Louvain method [5], and calculate the Normalized Mutual Information (NMI) between the detected sets.[5] The Comms feature needs a broad and deep view of the network, because it needs detailed knowledge about the entire network.

4. PROPOSED METHOD: MAXOUTPROBE

We propose the MaxOutProbe algorithm as a solution to the active graph probing problem. MaxOutProbe relies on the intuition that a successful probing strategy is one that brings many new nodes into the sample.

In terms of Figure 2, we want MaxOutProbe to probe yellow nodes that are adjacent to many green nodes. The goal of MaxOutProbe is to predict which candidate probes (yellow nodes) are adjacent to many nodes outside of the sample (green nodes). The major challenge that MaxOutProbe faces is thus accurately predicting the number of nodes that a given node is adjacent to outside of the sample. We begin this section with an overview of MaxOutProbe, and then give specific sample-dependent details.

4.1 MaxOutProbe

For each node u in G_{samp} that was not already fully explored during sampling, MaxOutProbe estimates

[4]A k-core of graph G is the largest induced subgraph of G where each node is connected to at least k other nodes, and the fringe contains all other nodes.

[5]NMI is a measure of cluster similarity, and identical clusterings have an NMI of 1.

d_u^{out}, the number of u's neighbors that lie outside of G_{samp}, as follows:

$$d_u^{out} = d_u - d_u^{in} = d_u - d_u^{known} - d_u^{unknown} \quad (1)$$

where:

- d_u is u's true degree in G_{orig}. This quantity is not known, and must be estimated.
- d_u^{in} is the number of nodes in G_{samp} that u is adjacent to in G_{orig}. This includes:
  - d_u^{known}, the number of nodes in G_{samp} that we already know to be adjacent to u. This quantity can be directly calculated.
  - $d_u^{unknown}$, the number of nodes in G_{samp} that u is connected to in G_{orig}, but not in G_{samp} (i.e., the connections to u that have not been observed). This quantity must be estimated.

Once d_u^{out} has been calculated, MaxOutProbe selects the b nodes with the highest scores (where b is the probing budget).

To estimate $d_u^{unknown}$, MaxOutProbe uses the knowledge that real social networks tend to exhibit high clustering (i.e., nodes tend to connect to friends-of-friends). More precisely, let w be an unexplored node that is two hops away from an unexplored node u, such that w and u are not connected in G_{samp} (they may be connected in G_{orig}). w forms an open wedge with u, because w and u share at least one neighbor. Let W_u be the number of such nodes w; these are the nodes in G_{samp} to which u is most likely connected.

Suppose MaxOutProbe estimated the transitivity C of the network, which defines the fraction of wedges that are closed triangles (that is, the ratio of 3 times the number of triangles in the network to the number of length-2 paths in the network). Given this value, MaxOutProbe estimates that u is connected to $C \times W_u$ nodes in G_{samp}, in addition to its known neighbors in G_{samp}.[6] Putting this all together, MaxOutProbe obtains the estimate:

$$d_u^{out} = d_u - d_u^{known} - d_u^{unknown} = d_u - d_u^{known} - (C \times W_u) \quad (2)$$

d_u^{known} and W_u can be calculated exactly from G_{samp}. The challenge thus lies in estimating d_u and C. The methods used for estimating these quantities are specific to the sampling method used to generate G_{samp}, and are discussed in the next section.

Figure 3 illustrates this process, where we are interested in node u. In G_{samp}, node u is adjacent to several red nodes, whose links are known. u may also be adjacent to some yellow nodes, but these links are unknown because neither u nor these nodes have been fully explored. Finally, u is adjacent to some green nodes, which are not present in G_{samp}. These links are also unknown. MaxOutProbe estimates d_u^{out}, the number of green nodes to which u is adjacent (here, d_u^{out} is 3). To estimate this value, it uses d_u, u's estimated true degree in G_{orig} (here, the true value of d_u is 8), and subtracts d_u^{in}, the estimated number of nodes within G_{samp} to which u is adjacent. d_u^{in} includes neighbors that MaxOutProbe is aware of (the three red nodes) as well as neighbors that it is unaware of (the two yellow nodes). To estimate the number of yellow nodes that u is adjacent to, MaxOutProbe uses the graph's transitivity. u participates in wedges with the yellow nodes at the top of Figure 3. Suppose that the transitivity of G_{orig} is $\frac{1}{3}$. MaxOutProbe then estimates that u is adjacent to $\frac{1}{3}$ of these two-hop neighbors. By combining all of this, MaxOutProbe predicts the number of neighbors that u has outside of G_{samp}, and selects the b nodes with the highest d^{out}.

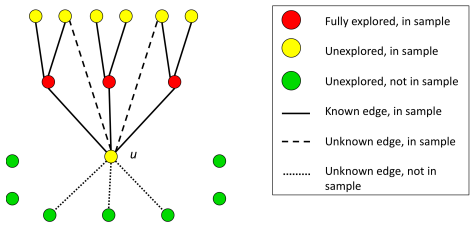

Figure 3: Intuition behind MaxOutProbe. The goal is to estimate the number of green out-of-sample nodes that u is adjacent to.

4.2 Estimating d_u and C

MaxOutProbe must accurately estimate each node's true degree d_u as well as the graph's transitivity C. In this section, we describe how MaxOutProbe makes these estimates for the five popular sampling methods from Section 3.

As before, u is a node from G_{samp} that was not fully explored during the sampling process. Let f_E and f_N, respectively, denote the fraction of edges and nodes from G_{orig} that are present in G_{samp}.[7]

4.2.1 *Estimations for Random Node Sample*

Estimating degree d_u: The Random Node sampling method randomly selects f_N fraction of nodes from G_{orig} for exploration, and learns all neighbors of the selected nodes. Thus, if node u has observed degree d_u^{known} in G_{samp}, MaxOutProbe estimates its true degree as

[6] Although u's individual clustering coefficient would be more valuable here than the global transitivity C, we have incomplete information about u, and thus use C as an approximation for the per-node clustering coefficients.

[7] We will give examples using real networks, described in Section 5.1. Section 6.2 provides experimental results regarding the quality of MaxOutProbe's estimates for d_u and C. We prove that the estimates are unbiased for some methods.

$\frac{1}{f_N} \times d_u^{known}$. E.g., if u is adjacent to 5 nodes in G_{samp}, and 10% of the nodes from G_{orig} were selected during sampling, then MaxOutProbe estimates u's true degree d_u as 50. This estimation works well for high degree nodes, but is less accurate for low degree nodes. However, as observed in Section 5, note that low degree nodes are unlikely to be useful probes, so this method is accurate where it needs to be.

Claim: The estimator for d_u is unbiased.

Proof: Let $\hat{d_u}$ represent the estimator for d_u. We must show that for each u, $E(\hat{d_u}) = d_u$. Recall that f_N fraction of nodes from G were selected uniformly at random during sampling to produce G_{samp}; these nodes plus their neighbors constitute G_{samp}.

Consider a node u that has not been explored during the sampling process, but is present in the sample G_{samp} (that is, u is adjacent to a node that was explored during sampling). Node u has d_u neighbors in G, and because f_N of the nodes from G were sampled uniformly at random to produce G_{samp}, $E(d_u^{known}) = f_N d_u$. Thus, $\frac{1}{f_N} E(d_u^{known}) = d_u$, so $E(\frac{1}{f_N} d_u^{known}) = d_u$. $\hat{d_u} = \frac{1}{f_N} d_u^{known}$, so $E(\hat{d_u}) = d_u$, so $\hat{d_u}$ is an unbiased estimator for d_u. □

Estimating transitivity C: To find C, MaxOutProbe estimates the number of wedges (2-paths) and triangles in G_{orig}.

Suppose that a triangle (x, y, z) exists in G_{orig}. What is the probability that it will be preserved in G_{samp}? In order for us to observe all edges $(x, y), (y, z)$, and (x, z) in G_{samp}, then at least one node from each of the three member edges must be fully explored during sampling. Thus, the triangle will be preserved if and only if at least two of the three nodes are selected. We can write the probability p_T that this occurs as:

$$p_T = 3f_N^2(1 - f_N) + f_N^3 \tag{3}$$

In other words, p_T is the probability that exactly two of the three nodes are selected, plus the probability that all three are selected. Thus, by multiplying the observed number of triangles in G_{samp} by $\frac{1}{p_T}$, MaxOutProbe estimates the number of triangles T in G_{orig}.

Now we calculate the probability that a length-2 path (x, y, z) from G_{orig} is preserved in G_{samp} (x may or may not be connected to z). To observe this path in G_{samp}, either x and z must *both* be probed, or y must be probed. The probability p_W of this occurring is as follows:

$$p_W = f_N^3 + 3(f_N^2(1 - f_N)) + f_N(1 - f_N)^2 \tag{4}$$

This is the probability that all three nodes are probed plus the probability that two nodes are probed plus the probability that just y is probed. MaxOutProbe then multiplies the observed number of length-2 paths from G_{samp} by $\frac{1}{p_W}$ to estimate the number of wedges W in G_{orig}.

G_{orig}'s transitivity C is then estimated as $\frac{3T}{W}$.

Claim: The estimator for C is unbiased.

Proof: Let $\hat{C}$ represent the estimator for C. We must show that $E(\hat{C}) = C$. f_N fraction of nodes from G were selected uniformly at random during sampling to produce G_{samp}.

Suppose that wedge (x, y, z) from G is present in G_{samp}. Suppose that (x, y, z) is a closed triangle in G (so edge (x, z) is present in G): what is the probability that edge (x, z) is present in G_{samp}, so (x, y, z) is a closed triangle in G_{samp}? Because we assume that (x, y, z) is present in G_{samp}, some of those three nodes must have been selected during the sampling process.

There are three possibilities: (1) all three of x, y, z were selected, (2) exactly two of x, y, z were selected, or (3) only y was selected (if only x or only z was selected, then either edge (y, z) or edge (x, z) would be absent from G_{samp}.

Possibility (1) occurs with probability f_N^3. Possibility (2) occurs with probability $f_N^2(1 - f_N)$. Possibility (3) occurs with probability $f_N(1 - f_N)^2$. In possibilities (1) and (2), edge (x, z) will certainly be present in G_{samp}. In possibility (3), edge (x, z) will not be present in G_{samp}. Thus, given that wedge (x, y, z) is present in G_{samp}, the probability P_{closed} that edge (x, z) is also present in G_{samp} is as follows:

$$P_{closed} = \frac{f_N^3 + ef_N^2(1 - f_N)}{f_N^3 + 3f_N^2(1 - f_N) + f_N(1 - f_N)^2} \tag{5}$$

Thus, of the wedges that are closed wedges in G, we expect that P_{closed} fraction of the wedges present in G_{samp} to be closed in G_{samp}. In other words, $E(P_{closed})C = C_{samp}$.

Note that the definition of $\hat{C}$ above is simply $\frac{1}{P_{closed}} C_{samp}$, so $E(\hat{C}) = C$, so $E(\hat{C})$ is an unbiased estimator of C. □

The results above show that the estimates are correct in expectation, but do not provide concentration bounds. We can prove such results that bound the errors in expectations by applying Hoeffding's inequality [12]. While these bounds may be weak for each individual entry, they can provide the theoretical basis for showing that we will be able to identify all large degrees vertices, and the degree of vertex that we predict to have a large degree, will not be too small. We are not including this analysis here due to space constraints, but the principles of Theorem 6.2 in [26], will apply to our application.

4.2.2 Estimations for Random Edge Sample

Estimating degree d_u: In this sampling method, f_E fraction of edges are sampled uniformly at random. Thus, to estimate a node's true degree d_u, MaxOutProbe simply multiplies its observed degree d_u^{known} by $\frac{1}{f_E}$. As with the Random Node sampling method, this estima-

tion is inaccurate for low degree nodes, but these nodes are unlikely to be chosen as probes, and so inaccuracy here does not affect the final probe selections.

Claim: The estimator for d_u is unbiased.

Proof: As before, let $\hat{d_u}$ represent the estimator for d_u, and we must show that for each u, $E(\hat{d_u}) = d_u$. G_{samp} consists of f_E of the edges from G selected uniformly at random.

Consider a node u in G_{samp}, with true degree d_u. Because f_E of the edges from G were sampled uniformly at random, $E(d_u^{known}) = f_E d_u$. Thus, $E(\frac{1}{f_E} d_u^{known}) = d_u$. This first term is simply $\hat{d_u}$, so $\hat{d_u}$ is an unbiased estimator for d_u. □

Estimating transitivity C: MaxOutProbe first calculates the transitivity C_{samp} of G_{samp}. Consider a length-2 path (x, y, z) in G that is present in G_{samp}. If that path is a closed triangle in G (i.e., x is connected to z), there is a f_E probability that the path will be a closed triangle in G_{samp} (i.e., with f_E probability, edge (x, z) is present in G_{samp}). Thus, MaxOutProbe estimates $C = \frac{1}{f_E} C_{samp}$.

Claim: The estimator for C is unbiased.

Proof: As before, let $\hat{C}$ represent the estimator for C, and we must show that $E(\hat{C}) = C$. G_{samp} consists of f_E of the edges from G selected uniformly at random.

C, the transitivity of G_{orig}, is the fraction of wedges (length-2 paths of nodes (x, y, z)) that are closed (x is connected to z). Suppose that edge (x, z) exists in G; then, independently of wedge (x, y, z) being present in G_{samp}, there is f_E probability that it is present in G_{samp}. Thus, of the wedges from G that are present in G_{samp}, we expect that $f_E C$ fraction of these wedges are closed in G_{samp} (C is the probability that it was closed in G, and f_E is the probability that it remained closed in G_{samp}). If C_{samp} is the transitivity of G_{samp}, then $E(C_{samp}) = f_E C$, so $E(\frac{1}{f_E} C_{samp}) = C$. The first term is $E(\hat{C})$, so the $\hat{C}$ is an unbiased estimator for C. □

The bounds discussed above also apply here.

4.2.3 Estimations for Random Walk and Random Walk w/ Jump Samples

Estimating degree d_u: We empirically observe that for the real networks, there exists some scale factor f_E^G such that multiplying u's sample degree by $\frac{1}{f_E^G}$ gives a good approximation of d_u. For example, Figure 4 shows the sample degree of each node in G_{samp} vs. its true degree in G_{orig} for a Random Walk sample from the Yahoo IM network. The sample degree and true degree are highly correlated, particularly for high degree nodes (over all nodes, a Pearson's correlation of 0.8). Here, f_E^G is slightly less than $f_E = 10\%$.

Thus, if MaxOutProbe can accurately estimate f_E^G, then it can get a good estimate of d_u, particularly for high degree nodes. To estimate f_E^G, MaxOutProbe uses a small amount of the probing budget b to probe nodes that have a high degree in G_{samp}.[8] MaxOutProbe then learns the true degrees of these nodes, and has already observed their degrees in G_{samp}. By taking the ratio of these values, MaxOutProbe can obtain an estimate of f_E^G.

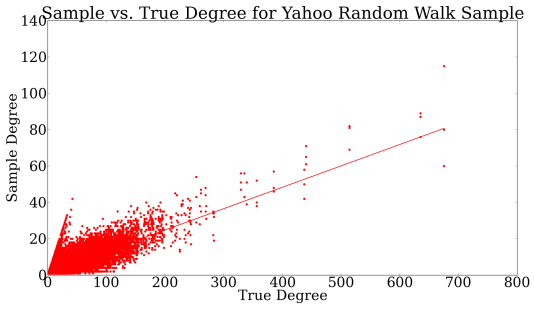

Figure 4: True vs. observed degrees for Random Walk sample of Yahoo network. Observe that there is a high correlation between a node's degree in G_{samp} and its degree in G_{orig}.

For the remaining unprobed nodes, MaxOutProbe estimates d_u by multiplying the observed sample degrees by f_E^G.

Estimating transitivity C: To estimate the transitivity C of G_{orig}, MaxOutProbe again uses the scale factor f_E^G. Suppose that a length-2 path (x, y, z) from G_{orig} has been preserved in G_{samp}. Suppose further that (x, y, z) is a triangle in G_{orig}, so x is connected to z. The probability that edge (x, z) was also preserved in G_{samp} is f_E^G. The chance that MaxOutProbe will observe (x, y, z) as a triangle rather than an open wedge is f_E^G, and so the transitivity of G_{samp} will be approximately a factor of f_E^G less than that of G_{orig}. Thus, to estimate the transitivity of G_{orig}, MaxOutProbe simply multiplies the transitivity of G_{samp} by f_E^G.

4.2.4 Estimations for BFS Sample

Estimating degree d_u: In the case of a BFS sample, all nodes that are potential candidates for probing (i.e., those that have not yet been fully explored during sampling) are on the fringe, or outermost layer, of the BFS. These nodes are adjacent to sampled nodes that are one level in from the fringe. To estimate the degree of these nodes, MaxOutProbe employs a similar strategy as with the random walk samples, and calculates a scale factor f_E^G. For a node u in the sample, MaxOutProbe estimates its true degree d_u by multiplying its observed sample degree by f_E^G.

To estimate f_E^G, MaxOutProbe examines the nodes

[8] MaxOutProbe uses between 1% and 10% of the probing budget on these initial probes; full details are available in Section 5.

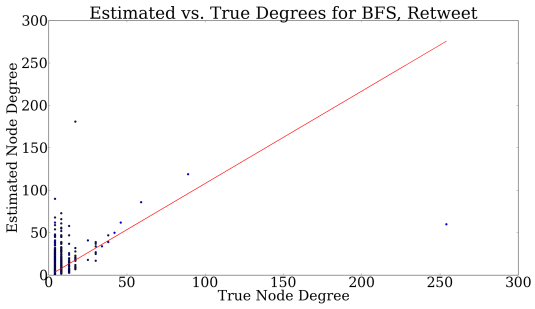

Figure 5: True vs. estimated values of d_u for nodes from a BFS sample of the Retweets network. Observe that the slope of the line is close to 1, indicating that the estimates are accurate on average.

that are one level deeper into the BFS, adjacent to at least one fringe node. Let L be the set of such nodes. Suppose that each node v in L is adjacent to $d_v^{BFS\_in}$ nodes deeper within the BFS, and has a true degree of d_v. Because MaxOutProbe has full information about the nodes in L, it can use them to approximate degrees of the fringe nodes. In particular, MaxOutProbe calculates the scale factor f_E^G by taking the average of the ratios of d_v to $d_v^{BFS\_in}$ for each $v \in L$.

Then, like with Random Walk sampling, for each candidate probe u, MaxOutProbe simply multiplies its degree in G_{samp} by f_E^G to estimate its true degree d_u.

Estimating transitivity C: A BFS sample gives us a complete, but biased, view of part of the network. The nodes in the sample are concentrated in part of the network and, with the exception of nodes in the fringe of the BFS, MaxOutProbe is aware of all connections between the sampled nodes. It calculates the transitivity of the subgraph obtained by removing the fringe nodes, and uses this as an estimate for the global transitivity C.

5. EXPERIMENTS

In our experiments, we demonstrate that MaxOutProbe outperforms a variety of baseline algorithms with respect to several different features, across different sampling methods.

We are interested in answering the following:

1. Is MaxOutProbe significantly better than the baseline probing strategies?
2. How much improvement does MaxOutProbe provide over baseline probing strategies?
3. How much does the quality of the estimations affect the performance of MaxOutProbe?

5.1 Datasets

We consider seven network datasets, described in Table 1. Five are communications networks, and the other two are similarity networks. In the similarity networks, each node represents a product (for Amazon, nodes are books, and for Youtube, nodes are videos), and a link between two products indicates that users have frequently purchased/watched both products.

Type	Network	Nodes	Edges	Transitivity	Comps
Communications	Enron	84K	326K	0.08	950
	Yahoo	100K	595K	0.08	360
	Replies	261K	309K	0.002	11,315
	Retweets	40K	46K	0.03	3,896
	LBL	3K	9K	0.005	9
Product Similarity	Amazon	270K	741K	0.21	3840
	Youtube	167K	1M	0.007	1

Table 1: Statistics for the network datasets that we consider: number of nodes and edges, transitivity, and number of connected components.

5.2 Baseline Approaches

We consider a variety of baseline heuristics, described in Table 2, for selecting which nodes to probe, each with a simple scoring function for selecting nodes.

These strategies each rank all of the nodes in the sample graph G_{samp} (except those that have already been fully explored, if known). These strategies are 'Exploit' strategies, which strategically select nodes, or an 'Explore' strategy, which randomly selects nodes.

The degree-based methods rely on the intuition that high-degree nodes in G_{samp} are connected to many nodes outside of G_{samp}. The structural hole methods target nodes that "fill" structural holes, thus connecting different parts of the graph [7]. On a similar intuition, the CrossComm strategy selects nodes on the border of two communities in the hopes.

The Random probing strategy selects random nodes from within G_{samp} for probing.

5.3 Experimental Setup

Recall that our primary goal is to demonstrate that MaxOutProbe outperforms the baseline probing strategies over a variety of features and sampling methods.

For each network from Section 5.1, we generate 20 samples for each of the five sampling methods described in Section 3.1. Each sample contains 10% of the edges from the original network. For each probing strategy, we conduct probes at budgets $b \in \{1\%, 2\%, 3\%, 4\%, 5\%, 10\%\}$ of the number of nodes in G_{orig}.

After conducting probes on a sample graph G_{samp}, we obtain an augmented sample graph G'_{samp}. We evaluate the quality of G'_{samp} with respect to the features described in Section 3.2, using the feature-specific evaluation functions also discussed in Section 3.2.

5.4 Evaluation

Our evaluation is intended to compare the performance of MaxOutProbe to the baseline strategies,

Category	Sub-Category	Strategy Names	Description
Exploit	Degree	**HighDeg, LowDeg**	Select the highest or lowest degree nodes.
	Structural Hole	**HighDisp, LowDisp**	Select the highest or lowest dispersion nodes.
		CrossComm	Pick nodes with the highest fraction of neighbors outside of their community (as identified with the Louvain method [5]).
	Clustering	**HighCC, LowCC**	Select the highest or lowest clustering coefficient nodes.
Explore		**Random**	Randomly select nodes from the sample.

Table 2: Baseline probing strategies. We categorize strategies as explore or exploit, and further subdivide as degree-based, structural hole-based, or random. Dispersion is an edge-based measure of how well a node's neighbors are connected to each other [3]. For each node, we average the dispersion of each of its adjacent edges.

analyze the quality of MaxOutProbe's estimates of d_u and C, and determine how much the error in the estimates affects the final outcome of MaxOutProbe.

To compare strategies, for a combination of network, sampling method, and feature, we do the following:

For each sampling method, we generate 20 samples: $G_{samp,1}, ..., G_{samp,20}$. For each sample, we identify probes using MaxOutProbe and the baseline strategies at all 6 probing budgets. For each probing strategy and each sample graph, we obtain an augmented sample graph, which we score using the feature-specific scoring functions described in Section 3. Let $S^i_{b,strat}$ represent the score obtained by using probing strategy *strat* on sample graph $G_{samp,i}$ at probing budget b.

Then for each pair of strategies *strat*1 and *strat*2, using all budgets b, we conduct a paired t-test to compare the values of $S^i_{b,strat1}$ and $S^i_{b,strat2}$ to ascertain whether *strat*1 is significantly better than *strat*2 at the $p = 0.01$ level. Additionally, for each budget b, we compare the means of the $S^i_{b,strat}$ scores (across all samples $G_{samp,i}$) to evaluate their differences in success.

6. RESULTS AND ANALYSIS

We first demonstrate that MaxOutProbe is significantly better than the baseline probing strategies at the various feature estimation tasks that we consider. We then evaluate the quality of estimates for MaxOutProbe, and analyze effects of estimation errors.

6.1 Comparison of MaxOutProbe to Baselines

Across a variety of networks and sampling methods, MaxOutProbe is typically significantly better than the baseline strategies at the feature tasks that we study. We conduct the paired t-test as described above, and present results in Table 3 at $p = 0.01$. In general, MaxOutProbe significantly outperforms the best baseline strategy. The results for the Random Node, Random Edge, Random Walk, and Random Walk w/Jump sampling methods are especially good. For these sampling methods, MaxOutProbe is particularly successful for the LCC, PageRank, and Core-Peri-2 features.

By how much does MaxOutProbe outperform the

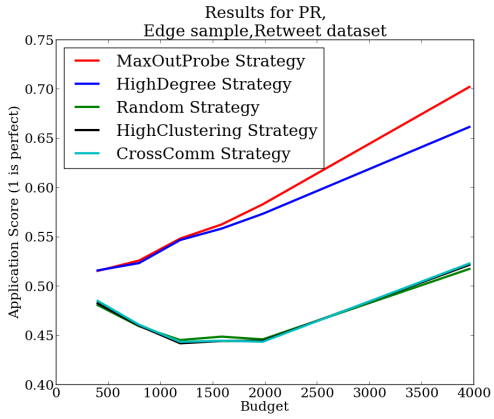

Figure 6: Average results for PageRank feature on Retweet network for Random Edge samples. MaxOutProbe is the best strategy for larger budgets.

baselines? This amount varies by budget, and is also dependent on the initial sample: if G_{samp} itself receives a high score with respect to the feature, then it will be difficult for any strategy to greatly improve the score.

Due to these issues, we cannot aggregate results over networks and budgets. Thus, we present one plot for each feature, showing results for one network and sampling method, averaged across the 20 samples. Figures 1 and 6 - 8 correspond to the LCC, PageRank, Community Detection, and Core-Periphery features. Rather than showing results for each baseline strategy, we selected the best-performing baseline from Table 2.

Plots are representative of other results for the feature, subject to Table 3. For example, Figure 1 shows that MaxOutProbe outperforms the baselines, and Table 3 tells us that this is generally the case.

We now quantify MaxOutProbe's improvement over the High Degree strategy, which is typically the best baseline strategy, at a fixed budget. We set a budget, and for each sample graph G_{samp}, calculate s_h, the amount by which High Degree probing improved G_{samp} with respect to the feature, and s_m, the amount by which MaxOutProbe improved G_{samp}. We take

	Features				
Samp. Method	LCC	PageRank	Comms	2-C.-P.	4-C.-P.
RandNode	6/7	7/7 (tied on 1/7)	5/7 (tied on 2/7)	5/7 (tied on 1/7)	3/7 (tied on 1/7)
RandEdge	7/7	7/7	4/7 (tied on 2/7)	6/7	4/7
Random Walk	6/7 (tied on 1/7)	7/7 (tied on 2/7)	5/7 (tied on 2/7)	6/7 (tied on 1/7)	4/7 (tied on 1/7)
Random Walk w/Jump	7/7	7/7	5/7 (tied on 2/7)	7/7	3/7
BFS	5/7 (tied on 2/7)	5/7 (tied on 3/7)	3/7 (tied on 2/7)	3/7 (tied on 1/7)	3/7 (tied on 1/7)

Table 3: Number of networks, out of the 7 datasets that we considered, on which MaxOutProbe outperformed all baselines. 2-C-P and 4-C-P represent the 2-Core-Periphery and 4-Core-Periphery features. Comparison was done using a paired t-test at $p = 0.01$. In some cases, MaxOutProbe tied for highest with some other strategy. MaxOutProbe is consistently the best for the LCC, PageRank, and 2-Core-Periphery features.

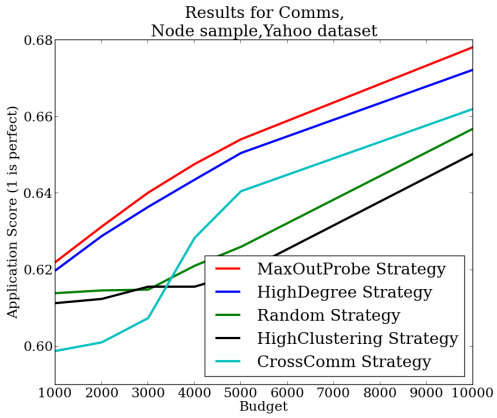

Figure 7: Average results for Comms feature on Yahoo network for Random Node samples. MaxOutProbe is the best strategy.

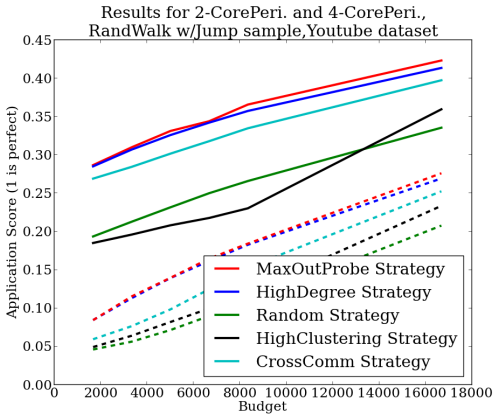

Figure 8: Average results for Core-Periphery feature on Youtube network for Random Walk w/ Jump samples. Solid lines correspond to the 2-CorePeriphery results, dashed lines to the 4-CorePeriphery results. MaxOutProbe is the best strategy at all budgets, but is similar to the High Degree strategy.

the ratio of s_m to s_h to see how much better MaxOutProbe is than High Degree probing. For example, suppose G_{samp} before probing receives a score of 0.5; and the sample after probing according to the High Degree strategy receives a score of 0.6; and the sample after probing according to MaxOutProbe receives a score of 0.65. MaxOutProbe is then a 50% improvement over High Degree probing (0.15 vs 0.10).

Table 4 contains the median improvement over all networks at a budget of 0.04. Individual results are often much higher (e.g., for PageRank on the LBL network with a Random Edge sample, MaxOutProbe improves on High Degree probing by 98%). These median differences are sometimes small, but they are statistically significant, and the improvement may be of critical importance in domains such as cybersecurity.

6.2 Quality of Estimates

MaxOutProbe estimates a node's degree d_u as well as the graph's transitivity C. We now examine the quality of the estimates, as described in Section 4.

6.2.1 *Estimating d_u*

We first calculate the quality of the degree estimates. Consider Figure 5, depicting unexplored nodes from a BFS sample from the Twitter Retweets network. The plot shows their true degree and the degree that MaxOutProbe estimated as described in Section 4.2.4. We fit a best fit line while forcing a y-intercept of 0. The quality of the estimate is high: 1.04 on average.

We calculate this value for each network and sampling method. Table 5 lists the slopes of the best fit lines: values close to 1 indicate a high quality estimate.

For the Random Node and Random Edge samples, for the Replies, Retweets, and Amazon networks, the ratios are at most 0.6, indicating that MaxOutProbe's estimates are poor. This occurs when the network has many very low degree nodes; as we noted in Sections 4.2.1 and 4.2.2, we expected MaxOutProbe to do poorly at estimating true degree for nodes with low degrees in the sample. Indeed, if we recalculate the values in Table 5 but only consider high degree nodes with true degree greater than 50, we see values very close to 1.

Samp. Method	Features				
	LCC	PR	Comms	2-C.-P.	4-C.-P.
RandNode	7%	5%	0%	4%	1%
RandEdge	2%	7%	1%	1%	1%
Rand Walk	3%	7%	1%	5%	1%
RW w/Jump	3%	5%	0%	2%	0%
BFS	10%	8%	0%	4%	0%

Table 4: Median improvement of MaxOutProbe over High Degree probing at $b = 0.04$ budget. Improvement is greatest for LCC, PageRank, and 2-Core-Peri.

Network	Degree Estimates				
	RandNode	RandEdge	RW	RWJ	BFS
Enron	0.9	1.0	0.9	1.0	0.8
Yahoo	0.7	0.8	0.9	0.7	1.4
Replies	0.4	0.6	1.1	0.9	1.0
Retweets	0.4	0.6	1.2	1.0	1.0
LBL	0.8	0.9	1.1	1.0	0.0
Amazon	0.4	0.6	0.9	0.7	1.1
Youtube	1.0	1.0	1.7	1.6	0.0

Table 5: Slope of best fit line comparing estimated degrees to true degrees. Observe that ratios are generally close to 1, indicating high quality estimates. In some cases, ratios are low: this occurs when the network contains many low degree nodes, which are difficult to get a good estimate for. See text for further explanation.

For example, for the Random Node samples, the Replies, Retweets, and Amazon networks have scores in Table 5 of 0.4, but for nodes with true degree greater than 50, we see ratios of 0.9, 1.0, and 0.8. Similarly, for the Random Edge samples, when we consider only high degree nodes, we see ratios of 0.9, 1.0, and 0.9, respectively.

As discussed earlier, the nodes that we wish to probe are generally those with high degrees in G_{orig}; thus, the poor estimates for low degree nodes do not affect the overall performance of MaxOutProbe.

For BFS samples, we see poor results on the LBL and Youtube networks. This is due to the hub-and-spoke structures of these networks: nearly every node has degree 1 in G_{samp}. Some of these nodes have high degree in G_{orig} but MaxOutProbe cannot determine that from the degree in G_{samp}.

6.2.2 *Estimating C*

We now perform a similar comparison for transitivity, and measure the ratio of estimated transitivity to true transitivity. Transitivity is difficult to estimate, as it can vary considerably by node [27]. Additionally, random walk samples tend to concentrate in more heavily clustered regions of the graph [22].

For Random Node sampling, most ratios are close to 1 (e.g., on the Enron network, the ratio of estimated to true transitivity is 0.8, and on Yahoo, it is 1.0). On some networks where clustering coefficient varies significantly by node, estimates are poorer (e.g., on Twitter Replies, MaxOutProbe overestimates by a factor of 5). We see similar results for Random Edge sampling: most ratios are close to 1, but MaxOutProbe sometimes overestimates substantially.

For Random Walk sampling methods, MaxOutProbe overestimates the true transitivity. This occurs because, as observed before, these samples possess higher transitivity than the complete graph. For the random walk sampling methods, MaxOutProbe typically overestimates transitivity by a factor of 1.5 to 3.

For the BFS samples, MaxOutProbe typically estimates transitivity within an order of magnitude. For the LBL and Youtube networks, which have a strong hub-and-spoke structure, the BFS sample contains mostly fringe nodes; when MaxOutProbe estimates transitivity by looking only at the fully explored nodes, it has very little information. On the LBL network, the BFS sample has a transitivity of 0! These two networks both have very low transitivity to begin with, so MaxOutProbe's low estimation is close to the truth. For other networks, MaxOutProbe obtain reasonable results (e.g., on Twitter Replies, MaxOutProbe's estimate has an excellent ratio of 0.99 to the true transitivity).

6.2.3 *Effect of Estimates on* MaxOutProbe

How does estimation error affect the performance of MaxOutProbe? To answer this, we create three 'cheat' strategies that use the true values of these quantities. In the first strategy, the true d_u values are known, in the second, C is known, and in the third, both are known.

Table 6 presents results for the PageRank feature on the Enron network. We present median scores for each strategy, at budget $b = 0.03$, over 20 graph samples. These results are representative of other networks.

We note several important observations:

1. Accurate estimation of the true degree is critical: Note the relatively poor performance of the Cheat2 strategy, in which degrees are estimated.
2. Knowing the true transitivity is not helpful: The Cheat1 and Cheat3 perform similarly. This is likely because transitivity varies substantially by node. Because the initial G_{samp} graphs are a small portion of G_{orig}, local transitivity can be of more value than the global transitivity.
3. Even though MaxOutProbe does not know the true degrees, adding estimated transitivity dramatically improves the results. Adding the true, global transitivity (Cheat2 strategy) does not help.
4. BFS samples are an exception to the above observations. This is due to the structure of BFS samples: nodes on the fringe of the BFS are the least-well connected to the sample as a whole (if they were better-connected, they would have been

observed deeper inside the BFS). Because they are poorly connected to the sample in general, clustering plays a smaller role in predicting connections.

6.3 Discussion

Our experimental results lead to several conclusions:

1. MaxOutProbe is significantly better than High Degree probing, the best baseline probing strategy. It is especially successful for the LCC, PageRank, and 2-Core-Periphery features, with a 3% - 10% improvement over High Degree probing.
2. MaxOutProbe shows the most consistent improvements when the sample is generated using the Random Node, Random Edge, or Random Walk sampling methods. Results on BFS are strong, but less consistent, as shown in Table 3.
3. MaxOutProbe does not perform as well for the Comms and 4-Core-Peri features. For these features, getting more data about nodes within G_{samp} is useful, rather than adding new nodes.
4. The quality of the degree estimate d_u is critical, and correctly estimating C is of less importance.

7. RELATED WORK

Our work is most related to graph sampling and crawling, as well as active learning, and researchers have studied related problems from other domains.

Crawling and Sampling Graphs Previous literature has examined the study of community detection from graph samples−e.g., by using minimum spanning trees (MSTs) [28], or by expanding a graph sample [16]. Avrachenkov *et al.* [2] show how to use queries to locate high-degree nodes, and O'Brien and Sullivan show how to use local information to estimate the core number of a node [19]. Hanneke and Xing, as well as Kim and Leskovec, infer characteristics of a larger graph given a sample akin to our Random Node sample [11, 13].

Our work is also related to the crawling literature. Maiya and Berger-Wolf [17] study the problem of online sampling for centrality measures including PageRank [18]. Cho, et al. study the problem of determining which URLs to examine first in a web-crawl so that a larger portion of the network is visited [9]. In contrast, we study the problem of selecting which nodes from an existing sample one should probe, rather than constructing a sample from scratch. Also, unlike most sampling methods in the literature, we are not attempting to down-sample a network to which we have complete access, but use a limited number of probes. Additionally, we select probes in a batch, rather than incremental, manner. This difference is critical in environments where getting data is time-consuming.

Active Learning. Our problem is related to active learning. E.g., Sheng *et al.* [25] consider the problem of when to get another label for elements in a class. Bilgic *et al.* [4] and Pfeiffer *et al.* [20, 21] studied active learning on networks for classifying nodes.

Related Work from Other Domains. Although we are the first to conduct a broad study in the problem of probing incomplete graphs, researchers from other domains, or those interested in specific network features, have studied related problems. Most similar to our work is that by Macskassy and Provost, who address the problem of identifying malicious entities in incomplete network data by gathering more information about those estimated to be the most suspicious [15].

Cohen, et al. propose a strategy for immunizing a population of individuals in an unobserved network by targeting neighbors of randomly selected nodes [10]. Shakkottai considers the problem of nodes in a sensor network attempting to find the source of information, given only local knowledge [24], and Ragoler et al. study the problem of determining the frequency at which sensor nodes should query their environment [23].

In addition to these problems, theoreticians have studied the structure of subgraphs through concepts such as monotone graph properties (see, e.g., Alon and Shapiro [1]). Charikar, et al. study the problem of estimating the value of a function over a set of inputs [8].

8. CONCLUSIONS

We introduced the problem of determining which nodes in an incomplete network to probe in order to learn the most information about the original network. We presented *MaxOutProbe*, a strategy for this problem.

We considered the Largest Connected Component, PageRank, Comms, and k-Core-Periphery features. We showed that MaxOutProbe outperforms baseline strategies at the task of identifying probes that are valuable for these features. We also analyzed the quality of the estimates produced by MaxOutProbe, and showed that estimating degree well is especially important.

Future work. We are working on several problems: How can we develop strategies for other probing scenarios (e,g., if we only obtain one edge per probe)? When do the gains from additional probes begin to diminish? What is the tradeoff between sample size and budget?

9. REFERENCES

[1] N. Alon and A. Shapira. Every montone graph property is testable. *SIAM J. Comput.*, 38(2):505–522, 2008.

[2] K. Avrachenkov, N. Litvak, L. O. Prokhorenkova, and E. Sayargulova. Quick detection of high-degree entities in large directed networks. In *ICDM*, pages 20–29, 2014.

[3] L. Backstrom and J. M. Kleinberg. Romantic partnerships and the dispersion of social ties: a

	PageRank Feature on the Enron Network Samples				
Probing Method	RandNode	RandEdge	Random Walk	Rand. Walk w/Jump	BFS
MaxOutProbe (estimate d_u and C)	0.87	0.89	0.72	0.95	0.39
Cheat1: True d_u, estimated C	0.96	0.91	0.76	0.87	0.45
Cheat2: True C, estimated d_u	0.50	0.56	0.54	0.64	0.46
Cheat3: True d_u and true C	0.95	0.92	0.76	0.87	0.52

Table 6: Effect of estimating d_u and C on MaxOutProbe. The numbers indicate median scores for each probing strategy, calculated at budget $b = 0.03$ and over 20 graph samples. The higher the scores, the better the performance. Accurate degree estimates are critical.

network analysis of relationship status on facebook. In *CSCW*, pages 831–841, 2014.

[4] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, pages 79–86, 2010.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, page 10008, 2008.

[6] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. Graph structure in the web. In *WWW*, pages 309–320, 2000.

[7] R. S. Burt. The social capital of structural holes. In M. F. Guillen, R. Collins, P. England, and M. Meyer, editors, *New Directions in Economic Sociology*. Russell Sage Foundation, 2002.

[8] M. Charikar, R. Fagin, V. Guruswami, J. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information. In *STOC*, pages 582–591, 2000.

[9] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. In *WWW*, pages 161–172, 1998.

[10] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Phys. Rev. Lett.*, 91(24):247901, 2003.

[11] S. Hanneke and E. P. Xing. Network completing and survey sampling. In *AISTATS*, pages 209–215, 2009.

[12] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Association*, 58(30):13–30, 1963.

[13] M. Kim and J. Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58, 2011.

[14] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636, 2006.

[15] S. A. Macskassy and F. Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *Int'l Conf. on Intell. Analysis*, 2005.

[16] A. S. Maiya and T. Berger-Wolf. Sampling community structure. In *WWW*, pages 701–710, 2010.

[17] A. S. Maiya and T. Y. Berger-Wolf. Online sampling of high centrality individuals in social networks. In *PAKDD*, pages 91–98, 2010.

[18] A. S. Maiya and T. Y. Berger-Wolf. Benefits of bias: towards better characterization of network sampling. In *KDD*, pages 105–113, 2011.

[19] M. P. O'Brien and B. D. Sullivan. Locally estimating core numbers. In *ICDM*, pages 460–469, 2014.

[20] J. J. Pfeiffer III, J. Neville, and P. N. Bennett. Active sampling of networks. In *MLG Workshop*, 2012.

[21] J. J. Pfeiffer III, J. Neville, and P. N. Bennett. Active exploration in networks: Using probabilistic relationships for learning and inference. In *CIKM*, pages 639–648, 2014.

[22] P. Pons and M. Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10(2):191–218, 2006.

[23] I. Ragoler, Y. Matias, and N. Aviram. Adaptive probing and communication in sensor networks. In *ADHOC-NOW*, volume 3158, pages 280–293, 2004.

[24] S. Shakkottai. Asymptotics of query strategies over a sensor network. In *INFOCOM*, pages 548–557, 2004.

[25] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622, 2008.

[26] D. Thompson, J. Bennett, C. Seshadhri, and A. Pinar. A provably-robust sampling method for generating colormaps of large data. In *LDAV*, pages 77–84, 2013.

[27] C. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM*, pages 608–617, 2008.

[28] J. Wu, X. Li, L. Jiao, X. Wang, and B. Sun. Minimum spanning trees for community detection. *Physica A*, 392(9):2265–2277, 2013.

A Multi-Armed Bandit Approach to Enriching Incomplete Graphs via Edge Probing

Sucheta Soundarajan & Tina Eliassi-Rad
Rutgers University
Email: {s.soundarajan, eliassi}@cs.rutgers.edu

Brian Gallagher
Lawrence Livermore National Laboratory
Email: bgallagher@llnl.gov

Ali Pinar
Sandia National Laboratories
Email: apinar@sandia.gov

***Abstract*—Relational data is often represented as a graph. In practice, the relationship data used to construct the graph is often incomplete; that is, the observed relationships are only a subset of the full set of relationships, and working with incomplete data can potentially skew analyses. Hoping to acquire the full data is also unrealistic, but we may be able to collect data more selectively.**

We introduce the *Active Edge Probing* problem, which asks the following: Suppose that one is given a sample of a larger graph and a budget to learn additional neighbors of nodes within the sample, with the goal of obtaining the most valuable information about the graph as a whole. Which nodes should be further explored? For example, a network administrator may have been given an incomplete map of her network, and can choose machines in which to run traceroutes, thus obtaining additional edges. Which machines should she select? We introduce ϵ-WGX, a multi-armed bandit-based algorithm for identifying which nodes in a graph sample should be probed. Our experiments compare ϵ-WGX to several baseline probing algorithms on four real network datasets using samples generated by four popular sampling methods. We consider two reward functions: (1) bringing in new nodes into the sample and (2) closing triangles within the sample, and show that ϵ-WGX significantly improves over random probing. For example, averaged over all sample types, at the task of closing triangles within the sample, ϵ-WGX improves over random probing by 29%.

I. Introduction

Most network analysis is conducted on existing incomplete samples of much larger complete graphs[1] (e.g., graphs collected over a short time period or produced by a crawling algorithm). For example, many researchers may obtain graphs from online dataset repositories. However, these graphs are often poor representations of the complete network. More complete data would lead to more accurate analyses, but data acquisition is costly. Given a query budget for identifying additional edges, how can one improve the graph sample so that it is a more accurate representation of the complete, fully observed network? This is a novel problem that is related to, but distinct from, topics such as graph sampling and crawling. Given the prevailing use of graph samples in the research literature, we believe that this problem is of considerable importance, even though it has been ignored.

We introduce and propose a solution to the *Active Edge Probing* problem, which we define as follows: Suppose that one obtains a sample G_{samp} of a larger graph G, without knowledge of or control over how that sample was generated.

[1]Throughout this paper, a 'complete' graph is a graph for which all data is available, as opposed to a clique structure in which all nodes are connected.

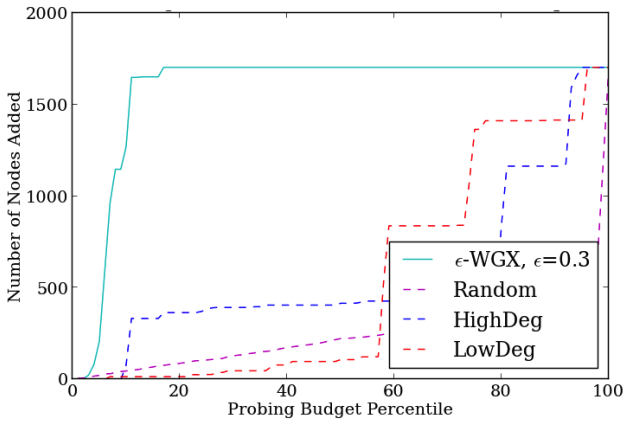

Fig. 1. Performance of the ϵ-WGX algorithm vs. baseline probing strategies on the FB-SocCir network using a sample produced by a BFS crawl, with the goal of maximizing the number of nodes in the sample, across a large range of probing budgets. ϵ-WGX outperforms the baselines by a very large margin.

Additionally, one has an edge budget b that can be used to *probe* nodes in G_{samp}, where each probe corresponds to learning one neighbor of the selected node (for example, given a sample of the Facebook network, one might decide to expand a portion of that network around some node, and thus inquire for information about that node). Which nodes should be probed to most effectively make use of the limited budget?

Depending on the analysis being conducted, the specific goal of this data acquisition will change. We consider two such goals: maximizing the number of nodes in the sample (i.e., adding edges leading out of sample), and closing triangles within the sample (i.e., adding edges within the sample).

Real applications of the Active Edge Probing problem abound. Consider a micro-loan company, which gets access to an applicant's Facebook account, and uses this social network to make credit decisions about the applicant. The company knows who the applicant's friends are but doesn't know if they are friends with each other, information that can be helpful for determining credit-worthiness. In this case, the company wishes to close triangles within the sample. Another example is of a researcher that has a sample of a Twitter retweet graph and wishes to bring new nodes and edges into his/her graph sample for more accurate community discovery.

To solve the Active Edge Probing problem, we present ϵ-WGX (Weighted Graph eXplore), a multi-armed bandit-based

strategy for identifying nodes for probing.

The key strength of ϵ-WGX is that it does not require any background information about the structure of the sample, knowledge of how the sample was created, details about the underlying network G, or intuition about which nodes should be good probing selections for the specific reward function. Although for specific datasets and reward functions, some baseline strategy might be more successful, ϵ-WGX is an excellent choice when one doesn't know which strategy to use.

We compare ϵ-WGX to several baseline probing strategies. We conduct experiments on four real networks, using samples generated by four popular sampling methods. Across sampling methods, networks, and reward functions, ϵ-WGX is always the best or second best probing strategy (e.g., for the task of maximizing the number of nodes brought into the sample, on samples generated by a BFS crawl, ϵ-WGX outperforms random probing by 20%); moreover, unlike these baseline strategies, an individual using ϵ-WGX requires no background knowledge about the behavior of the reward function or the structure of the network. For example, Figure 1 depicts the performance of ϵ-WGX vs. the three baseline strategies on a sample of a Facebook network (described in Section V-C) that was generated by a breadth-first-search crawl. In this figure, the goal of each strategy was to maximize the number of new nodes brought into the sample through probing. We considered a range of budgets varying from 0 probes to the case when every single node in the sample has been probed as many times as possible (at these extremes, all strategies naturally perform the same). We see that ϵ-WGX very quickly reaches the maximum performance, using a much smaller budget than the baseline strategies.

We argue and show that use of a simple baseline algorithm (such as probing the nodes with the highest degree in G_{samp}) is not appropriate for this problem: although the baseline may be successful for one particular sample or reward function, it will not be generally successful. For example, Figure 1 shows that the Low Degree probing strategy is more successful than the High Degree probing strategy at low probing budgets, but their positions reverse at higher probing budgets. In contrast, ϵ-WGX can adapt to the specific graph sample, reward function, and probing budget under consideration.

The contributions of our paper are as follows:

- We introduce the *Active Edge Probing* problem, which is of vital importance to a variety of applications in network analysis.
- We propose ϵ-WGX, a multi-armed bandit-based algorithm for strategically selecting nodes from a sample graph G_{samp} for probing.
- We present empirical evidence demonstrating that use of a baseline probing strategy is inappropriate for the Active Edge Probing problem, and show that for all considered sampling methods, networks, and reward functions, ϵ-WGX is always the best or second best strategy. This performance is statistically significant, measured at $p = 0.01$.

This paper is organized as follows: In Section II, we formally define the Active Edge Probing problem. We present preliminaries that are required for ϵ-WGX in Section III. Section IV presents the ϵ-WGX algorithm, and Section V describes our experimental set-up, including discussion of the reward functions. We present results in Section VI and discussion in Section VII. We then give an overview of related work in Section VIII, and conclude in Section IX.

II. Problem Statement

In the Active Edge Probing problem, one is given a graph G_{samp} that is a sample of a larger, unobserved graph G, and is interested in the results of some application (such as community detection) on G. To assist in obtaining accurate results, one can use a budget b to obtain additional information to supplement G_{samp}. This information is obtained as follows. For each unit of budget, one selects a node for probing, notifies the underlying network system or administrator of this choice, and is then given an edge adjacent to that node in the true graph G. This query might represent, for example, asking Twitter to return one follower of a given user, or using a traceroute to identify one of a node's neighbors.

In this paper, we investigate goal-specific data collection, and thus use a reward function, which assigns each returned edge a score corresponding to its value for the application of interest. We consider two reward functions, which correspond to the number of new nodes brought into the sample and the number of triangles formed in the sample, but any reward function is acceptable.

We investigage two types of queries. In the first, the user may specify the edges that she already knows about, and thus request a new edge that she is unaware of; i.e., if she queries a node multiple times, she is guaranteed to get a new neighbor each time. For example, if directly asking the administrators of an online social network such as Facebook or Twitter, one might send five queries and expect five unique neighbors. In the second, the user cannot specify edges that she already knows, and so may receive duplicate edges from multiple probes. This query type is relevant to domains in which one obtains neighbors of a node by observing the behavior of the underlying network, as opposed to performing a database query. For example, if one learns neighbors of a node through a traceroute, then there is no guarantee that a new neighbor will be observed with each query.

In both versions, we assume that after selecting a node for querying, a user is given a neighbor of that node selected uniformly at random (e.g., if we ask Twitter to tell us a neighbor of a node, all neighbors of that node are equally likely to be returned). In the first version, we assume that this selection is done *without replacement*; i.e., if we query a node multiple times, we are guaranteed to get a new neighbor each time (when no new neighbors remain, nothing is returned). In the second version, we assume that this selection is done *with replacement*; i.e., if we query a node multiple times, we may receive duplicate neighbors.

We refer to the first problem as the *AEP-NoReplacement* (AEP-NR) problem, and the second problem as the *AEP-Replacement* (AEP-R) problem.

After performing b probes on G_{samp}, one obtains an enhanced sample G'_{samp}, which contains the information from

G_{samp} as well as the information obtained from the probes. The goal of the Active Edge Probing problem is to maximize the overall score of the reward function on the graph G'_{samp}.

We assume that G is an unweighted, undirected graph (though our methods can easily be modified for directed or weighted graphs). We do not assume that the user knows the process by which G_{samp} was generated, or any information about G beyond the link structure in G_{samp}.

Challenges: A successful probing strategy must determine which nodes are likely to lead to high scores with respect to the reward function (e.g., which nodes should be probed to learn about the greatest number of nodes outside of the current sample). Additionally, for the AEP-R problem, in which queries may result in duplicate neighbors, the strategy must determine when to stop probing a node.

There are several difficulties in determining which probes are likely to provide important structural information. First and foremost, for a given task or application (such as maximizing the number of nodes in the enhanced sample G'_{samp}), it is not immediately clear how one can select nodes based only on their structural profile in G_{samp}. As we will see, even if one has observed that certain types of nodes (e.g., nodes with high degree in G_{samp}) are good candidates for probing, the success of such a strategy can vary across networks, sampling methods, and even probing budgets. Second, even if one could design a probing strategy that works well for one application, there is no reason to believe that this strategy would work for any other application.

The key challenge, then, is designing an algorithm that works across *networks, sampling methods, applications, and probing budgets.* We do not necessarily expect this algorithm to be the best performing strategy in every case (because as we have noted, it is unlikely that any single strategy is the best in all cases). However, we do expect that it should be a *safe* strategy: that is, it should consistently perform significantly better than random probing, and should frequently be close to the best strategy for the particular setting.

III. Preliminaries

A. Multi-Armed Bandits

The multi-armed bandit problem is motivated by a gambler attempting to choose which slot machine (aka 'bandit') to play from among a set of slot machines. Each machine will give the gambler a reward drawn from some distribution specific to that machine, and the goal of the gambler is to maximize this reward. This problem appears in a variety of settings, such as advertising, in which a company wishes to determine which advertisements are most effective.

A hallmark of solutions to the multi-armed bandit problem is the presence of *exploration*, or the selection of random machines, and *exploitation*, or playing machines that have been known to perform well in the past. The simplest solution to the multi-armed bandit problem is the ϵ-greedy algorithm [1]. In this method, one explores with probability ϵ (i.e., selects a machine uniformly at random), and exploits with probability $1-\epsilon$ (i.e., selects the machine with the highest average reward so far).

Since the development of the ϵ-greedy algorithm, a variety of algorithms have been proposed, including methods that use weighted exploration or contextual information [2]. In our work, we consider a variation of the ϵ-greedy method, and show that it works well for the AEP problem.

B. Predicting Population Size with Random Draws

A second major challenge in the AEP-R problem is determining when to stop probing a node. Suppose that we have probed a node k times, and seen w distinct neighbors and r duplicates ($k = w + r$). What is the estimated degree d of that node?

The problem of estimating a population size given multiple draws, and some number of distinct and duplicate observation, arises in combinators [3]. As shown in [3], the maximum likelihood estimate of d, given k, w, and r, is approximately:

$$\hat{d} = \frac{w + r}{m(s)}, \tag{1}$$

where $m(s)$ is the solution to $s = (1 - e^{-m})/m$. This method assumes that each edge is equally likely to be selected, but for the general case, we can use the Valiants' results [4].

C. Sampling Methods

The purpose of the AEP problem is to improve a graph sample in such a way as to produce the most valuable structural information. How are such samples made? In practice, samples are often generated in one of the following four ways:

- **Breadth-first search (BFS)**: The BFS sampling method begins with one node (often randomly selected), and iteratively adds all neighbors of nodes already in the sample. In our experiments, we fully explore (i.e., add all neighbors of) observed nodes, with the exception of nodes observed in the final iteration, on the fringe of the set. BFS crawls are a common way of exploring portions of the Web (e.g., in [5]).
- **Random Edge (RandEdge)**: The Random Edge sampling method selects a given fraction of edges uniformly at random from the entire network. For example, the Twitter firehose provides a 10% uniform sample over the set of tweets. If one is constructing a graph in which retweets constitute edges, then this data produces a sample similar to the Random Edge method.
- **Random Walk (RW)**: The Random Walk sampling method selects a random node as a starting point, then iteratively transitions to a random neighbor of the node that it is currently on. A single edge at a time is observed. This is also a common method for sampling large graphs [6].
- **Random Walk with Jump (RWJ)**: The Random Walk with Jump method is similar to the Random Walk sampling method, except that at each step, there is a 15% chance of transitioning to a random node in the graph.

In our experiments, we evaluate our algorithm on samples generated by these four methods.

IV. Proposed Method

We view the AEP problem as a version of the general multi-armed bandit problem: one may probe any of the nodes in the sample graph, and depending on the selection, obtains information of varying value for the feature of interest. For example, suppose that we wish to maximize the number of nodes in the enhanced sample G'_{samp}. In this case, it would be most valuable to probe nodes in G_{samp} that have a high fraction of neighbors outside of G'_{samp}. But how can one predict which nodes are likely to have a high fraction of neighbors outside of G_{samp}? We will see that it is not as simple as selecting nodes with high (or low) degree inside G_{samp}. Because successful probes may exhibit significant structural variance depending on application, network, and so on, multi-armed bandits are the ideal tool.

Our algorithm has the following strengths: (1) It can be used *without background knowledge of the network structure or reward function*. In other words, one need not know what type of node (e.g., high degree) is useful for the reward function. (2) It can *adapt* to the network and reward function: if probing some node is unsuccessful, it is unlikely to probe that node further. (3) It is *safe*: although it does not always outperform the best baseline for a given sample and reward function, it is significantly better than random, and typically comes close to matching the performance of the best baseline.

We initially present this algorithm as a solution to the AEP-NR problem; in Section IV-B, we describe how to modify it for the AEP-R problem.

A. ϵ-WGX

We introduce ϵ-WGX (Weighted Graph eXplore), an algorithm to solve the AEP problem.

ϵ-WGX requires that the user specify a reward function, which is used to evaluate the success of a probe (we consider two reward functions, described in Section V-B). For each node u, we maintain a reward vector R_u, which contains a list of rewards corresponding to the outcomes of the probes conducted on that node.

Additionally, the algorithm uses a weight function $W : u \to \mathbb{R}^+$ to guide probe selection. This function is described in Section IV-C, and is based on the reward function: nodes that have produced high rewards in the past have higher weights.

Finally, the algorithm requires a parameter ϵ between 0 and 1. In each step:

- Exploration: With probability ϵ, the algorithm randomly selects a node u from G_{samp} in accordance with the weight function.
- Exploitation: With probability $1 - \epsilon$, the algorithm selects u uniformly at random from among the set of nodes with the highest weights.

Once the probe is made, we update the reward vector R_u and the weight function. If u has no further neighbors to be learned, we remove it from future consideration.

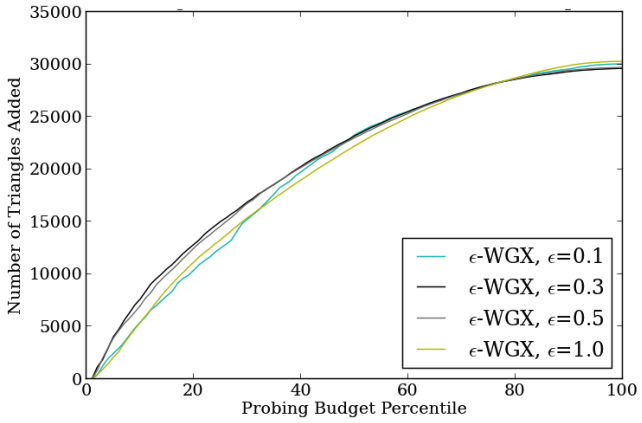

Fig. 2. Results for ϵ-WGX on a BFS sample from the FB-UGrad network (described in Section V-C). Observe that the performance is not sensitive to the choice of ϵ.

B. Solving the AEP-R Problem

For the AEP-R problem, when we perform a probe, we are not guaranteed to get a new neighbor of the selected node. We thus face the additional problem of determining when to stop probing a node. Suppose that for a node u we have conducted C_u probes in total, and have observed A_u distinct neighbors and B_u duplicates.

We devise the **Reduce Weight criterion**, in which the probability of probing a node u depends on its reward vector R_u, as well as the values of A_u and B_u, as follows:

Using A_u and B_u, we calculate p_u, the probability of observing a new neighbor when probing, using the MLE method described in Section III.[2] Using p_u, define a new weight function $W' : u \to \mathbb{R}^+$, which is defined by $W'(u) = p_u W(u)$. Then instead of selecting nodes according to the original weight function W, use the modified weight function W'.

C. Weight Function

Suppose a node u has been probed at least once. Then $W(u)$ is defined as the mean of the reward vector R_u. Suppose a node v has not yet been probed. Then $W(v)$ is the mean of all values $W(u)$, for nodes u that have been probed.[3]

D. Selection of ϵ

We considered a variery of ϵ values ranging from 0.1 (rarely explore) to 1.0 (always explore). Results are not very sensitive to ϵ (see Figure 2 for an example). Setting $\epsilon = 0.3$ seems to work well, and we present those results here.

[2]The MLE prediction for total degree uses only the values A_u and B_u. It is possible, though unlikely, that between the neighbors learned from probing as well as the neighbors that originally existed in G_{samp}, there are more known neighbors than predicted neighbors. Because it would be nonsensical to estimate p_u as a negative number, we estimate it as a very small number (here, 0.000001).

[3]Like with p_u, we add a small number (here, 0.1) to all values to ensure that even nodes with a mean observed reward of 0 have some probability of being probed.

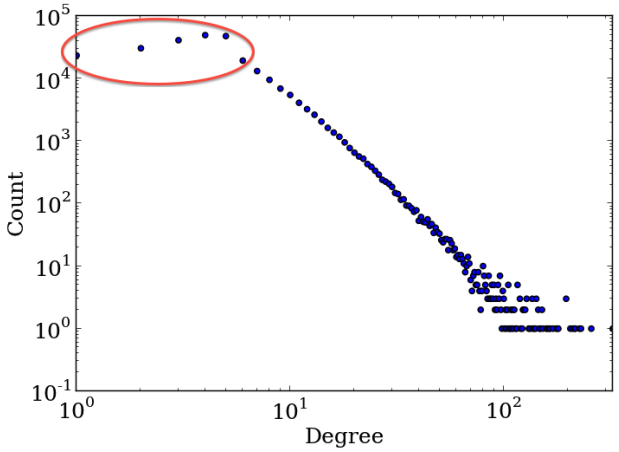

Fig. 3. Degree distribution for Amazon network (described in Section V-C). There are many nodes with low degree, but not overwhelmingly so.

E. Notes on Use of ϵ-WGX

It is important to note that ϵ-WGX is based fundamentally on the principles of exploitation and exploration; i.e., once it finds a successful node, it should be able to probe that node multiple times.

Consider the extreme case of a network where nearly every node has degree 1: when the algorithm randomly selects a node, it will learn the single edge, but will not be able to acquire new knowledge through additional probes. ϵ-WGX is most useful in cases where the typical node has a fairly high degree. See Figure 3 for an example of the degree distribution of such a network: although it does appear to follow a heavy-tailed distribution after some minimum x-value, degree-1 nodes are not an overwhelming majority.

V. Experimental Set-up

The purpose of our experiments are to demonstrate the following claim:

The ϵ-WGX algorithm consistently performs well. Although it does not always outperform the best baseline probing strategy for a particular network sample and task, the best baseline strategy varies across tasks, samples, and even probing budgets. Thus, the ϵ-WGX algorithm is a safe and reliable choice in cases when a user does not know which specific baseline probing strategy is appropriate for a given network, sampling method, task, and probing budget.

A. Experimental Overview

We assess the performance of our ϵ-WGX algorithm on four network datasets (see Section V-C), using samples generated by the four popular sampling methods (see Section III-C), and compare against relevant baseline probing strategies (see Section V-D) using two reward functions (see Section V-B).

For each network G, for each sampling method, we generate five samples, each containing 10% of the edges from the original network G. For each sample G_{samp}, we calculate b^{NR}_{max}, the maximum possible number of edges that can be learned (i.e., the number of edges adjacent to at least one node in G_{samp}, which are not already present in G_{samp}). For the AEP-R problem, in which one may obtain duplicate edges with multiple probes, it is theoretically possible to prove forever. Thus, we set an upper limit budget cutoff at $b^{R}_{max} = 3 \times b^{NR}_{max}$. We then consider budgets $b_1, ..., b_{100}$ corresponding to the 100 quantiles of the interval $(0, b^{NR}_{max}$ or $b^{R}_{max})$, as appropriate. For example, if $b^{NR}_{max} = 200$, we consider budgets $2, 4, 6, ..., 200$.[4] (Recall that each unit of budget corresponds to probing one node and learning one edge.)

For our ϵ-WGX algorithm, we set $\epsilon = 0.3$ (as described in Section IV-D), and the baseline strategies are described in Section V-D. We visually observe the behavior of the various strategies across the range of budgets and compare results at a single budget level; additionally, we use a Wilcoxon signed-rank test to evaluate whether the differences between algorithms are statistically significant.

B. Reward Functions

In this work, we consider two reward functions that are relevant for a variety of applications.

Let u be the node from the original sample graph G_{samp} that was probed, and let v be the neighbor of u that is obtained by probing u.

- The **Number of Nodes (NumNodes)** reward function has a value of 1 if node v is outside of G_{samp}, and 0 if v is within G_{samp}.
- The **Triadic Closure** reward function has a value of 1 if both u and v are within G_{samp}, edge (u, v) is not in G_{samp}, and u and v share at least one neighbor in G_{samp}, and a value of 0 otherwise. In other words, if u and v are 2-hops apart in G_{samp}, then edge (u, v) closes the triangle.

C. Datasets

We consider the following network datasets:

- **FB-Grad** This network represents a portion of the Facebook network corresponding to graduate students at a university. It contains 523 nodes and 3256 edges [7].
- **FB-UGrad** This network represents a portion of the Facebook network corresponding to undergraduate students at a university. It contains 1220 nodes and 43,208 edges [7].
- **FB-SocCir** This network represents a portion of the Facebook network collected by the Social Circles application. It contains 4039 nodes and 88,234 edges.[5]
- **Amazon** This network represents book co-purchases from Amazon.com. Each node represents a book, and two nodes are linked if one of the two books is frequently purchased with the other. It contains 270,347 nodes and 741,124 edges.[5]

[4]There is one exception to this: due to the larger size of the Amazon network, described in Section V-C, we only consider the 100 quantiles of the interval $[0, 0.02 \times b^{NR}_{max}]$.

[5]Obtained from snap.stanford.edu.

D. Baseline Probing Strategies

We considered a variety of baseline probing strategies based on structural information, and consistently saw that two performed the best: probing high degree nodes and probing low degree nodes. These two strategies probe nodes in descending or ascending order of degree until there are no more edges to be learned for a node. (For the AEP-R problem, we use a stopping criterion described below). We refer to these strategies as 'High Degree' (HighDeg) and 'Low Degree' (LowDeg) probing.

We additionally consider a random probe strategy ('Random'), which selects a random node for probing in each step. A node is eliminated from consideration once all of its edges have been learned.

Note that for the AEP-R problem, our Reduce Weight criterion described in Section IV-B cannot be applied to these baseline strategies. This is because the Reduce Weight criterion requires that each node be assigned a score, which is then multiplied by the probability of seeing a new edge when probed. These strategies *rank* the nodes, but do not score them.

For these methods, we use a simple rule to decide when to stop probing: if we have seen duplicate neighbors in more than half of a node's probes, we stop probing it.

E. Evaluation

Our evaluation method is specific to the reward function being used. For a given sample G_{samp}, once some number of b probes have been conducted, we score the enhanced sample G'_{samp} as follows:

- For the NumNodes reward function, we calculate the total number of nodes in G'_{samp}.
- For the Triadic Closure reward function, we calculate the total number of triangles in G'_{samp} between nodes that were originally in G_{samp}.

VI. Results

Recall that the goal of our experiments is to demonstrate that ϵ-WGX is a safe strategy: that is, it is significantly better than random probing, much better than the worst baseline, and frequently comparable to the best strategy. It is not reasonable to expect ϵ-WGX to outperform the best baseline, because the best baseline varies depending on the network, sampling method, reward function, and probing budget.

To this end, we present results in three parts:

1) We demonstrate that it is difficult to predict which baseline strategy is the best, as it varies between networks, sampling strategies, and probing budgets.
2) We discuss results for the AEP-NR problem, in which one is guaranteed to get a new edge with every probe. We show that the ϵ-WGX algorithm performs significantly better than random, and often even outperforms the best baseline strategy.
3) We discuss results for the AEP-R problem, in which one is not guaranteed a new edge with every probe. We again show that ϵ-WGX performs well.

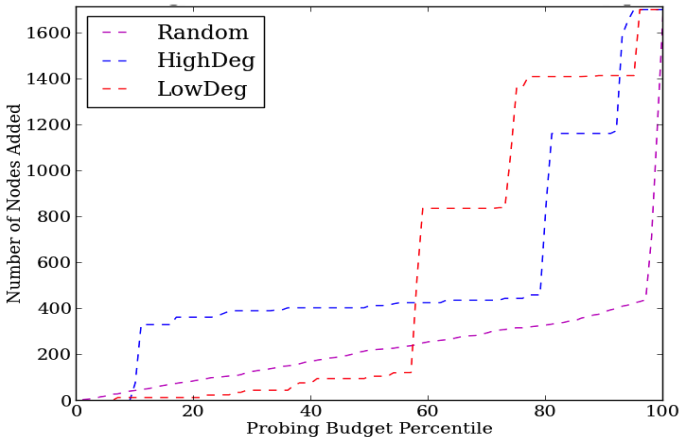

Fig. 4. Results of various baseline strategies on a BFS sample from the FB-SocCir network for the NumNodes reward function. Note that the choice of which baseline strategy is best depends on budget.

TABLE I. The best baseline strategy for various sampling methods and networks for the NumNodes reward function. There is little consistency. 'Varies' indicates that the best strategy is budget-dependent.

	Network			
SampleType	FB-Grad	FB-Ugrad	FB-SocCir	Amazon
BFS	LowDeg	LowDeg	Varies	LowDeg
Edge	LowDeg	LowDeg	HighDeg	HighDeg
RW	Varies	Varies	Varies	HighDeg
RWJ	Varies	Varies	HighDeg	HighDeg

A. Analysis of Baseline Strategies

We evaluate the three baseline strategies described in Section V-D on each sample over all considered probing budgets. We see that the best baseline can vary in surprising ways.

For example, see Figure 4, which shows the results of the baseline strategies over the range of probing budgets on a BFS sample from the FB-SocCir network, for the NumNodes reward function. Initially, High Degree probing is best, but later, Low Degree probing becomes the best.[6]

Table I lists the best baseline strategy across 5 samples for each sampling method and network, for the NumNodes reward function. Results are inconsistent: the best baseline strategy for one particular sample type and network is unlikely to be the best for a different sample type or network.

These results motivate ϵ-WGX, which adapts to the specific qualities of the graph sample and reward function.

B. Results for AEP-NR Problem

In the AEP-NR problem, one is guaranteed to get a new edge with every probe; thus one does not need to think about when to stop probing a node. We apply our ϵ-WGX algorithm to this problem, with excellent results.

[6]Both the curves for the High Degree and Low Degree strategies have three locations with a sharp jump in the number of nodes. This is because there are three nodes in the sample that are exceptionally good to probe (adjacent to many nodes outside the sample). These strategies encounter these nodes at different times.

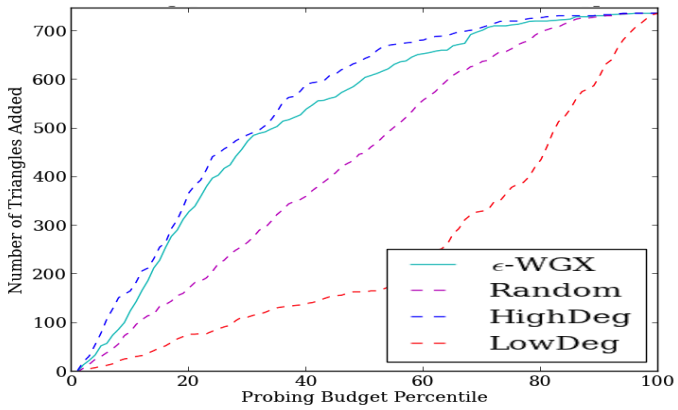

Fig. 5. Results of various strategies on a BFS sample from the FB-Grad network for the TriClose reward function. ϵ-WGX performs comparably with the best baseline. (Best viewed in color.)

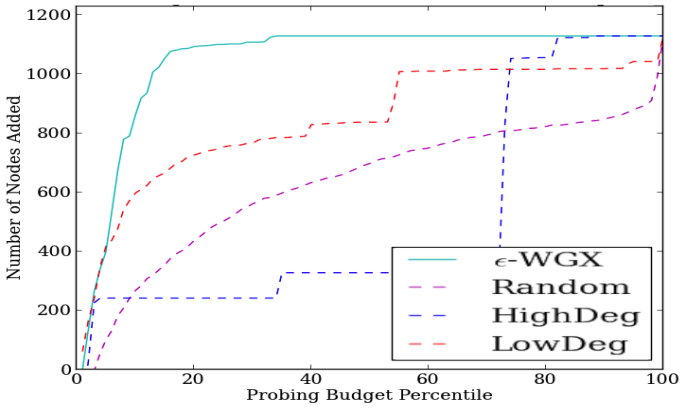

Fig. 6. Results of various strategies on a RW sample from the FB-SocCir network for the NumNodes reward function. ϵ-WGX outperforms the best baseline strategy. (Best viewed in color.)

Figure 5, presenting results for the ϵ-WGX algorithm on a BFS sample for the FB-Grad network across the range of budgets that we considered, is typical of other networks, sampling methods, and reward functions. Note that ϵ-WGX performs comparably with the High Degree probing strategy, which is the best baseline strategy (the differences here are not significant). Both perform much better than Random probing and Low Degree probing.

Figure 6 shows the same set of results, but on the FB-SocCir network. Here, ϵ-WGX is the best even at low budgets.

We present small versions of plots corresponding to one sample for each network and sampling type in Figure 7, for the NumNodes reward function. These plots generally fall into the same pattern as one of the plots discussed already. Takeaways for the TriClose reward function are similar, and to save space, we do not present them here.

To aggregate across multiple samples and networks, we perform the following: We fix a budget value b_{mid} at the midpoint of the range we consider (i.e., half of the maximum possible number of edges that can be learned). For each

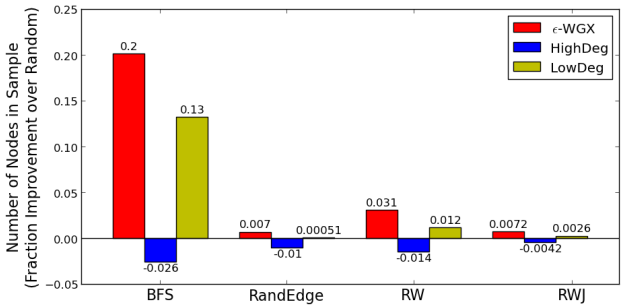

Fig. 8. Mean performance of ϵ-WGX vs baseline strategies, presented as fraction improvement over random, measuring number of nodes in the sample. For every sampling method, ϵ-WGX is the best probing strategy.

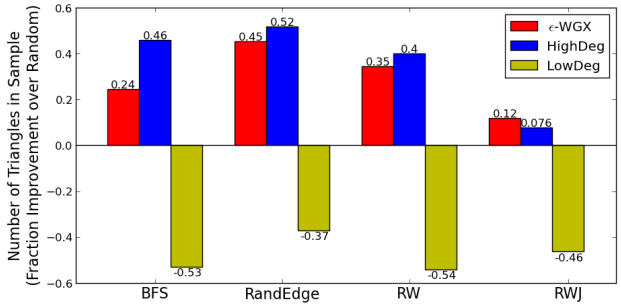

Fig. 9. Mean performance of ϵ-WGX vs baseline strategies, presented as fraction improvement over random, measuring number of triangles in the sample. For the RWJ sampling method, ϵ-WGX is the best strategy. For the others, it is second best.

sampling method, we perform b_{mid} probes according to ϵ-WGX and the three baseline strategies, over all five samples generated by that sampling method over each of the four networks. Figures 8 and 9 contain the means of the resulting scores for each probing strategy. We present these values as a fraction improvement over Random probing, so the Random probing strategy has a score of 0 and is not presented. For example, over all BFS samples from the four networks, ϵ-WGX had a mean 20% improvement over Random probing with respect to the NumNodes reward function.

It is interesting that the relative performance of the strategies varies with sample type. We discuss this in Section VI-D

To determine whether the differences are statistically significant, for each reward function and sampling method combination, we perform a Wilcoxon signed-rank test at the $p = 0.01$ level, aggregating all networks and budgets together. We compare ϵ-WGX to each of the three baseline strategies.

The differences shown in Figures 8 and Figures 9 are representative of the results over all budgets, and these differences are statistically significant. ϵ-WGX is the best strategy in all cases, with the following exceptions:

- For the TriClose reward function, High Degree probing is the best strategy for BFS, RandEdge, and RW samples. ϵ-WGX is second-best.

These results support our goal of demonstrating that the ϵ-

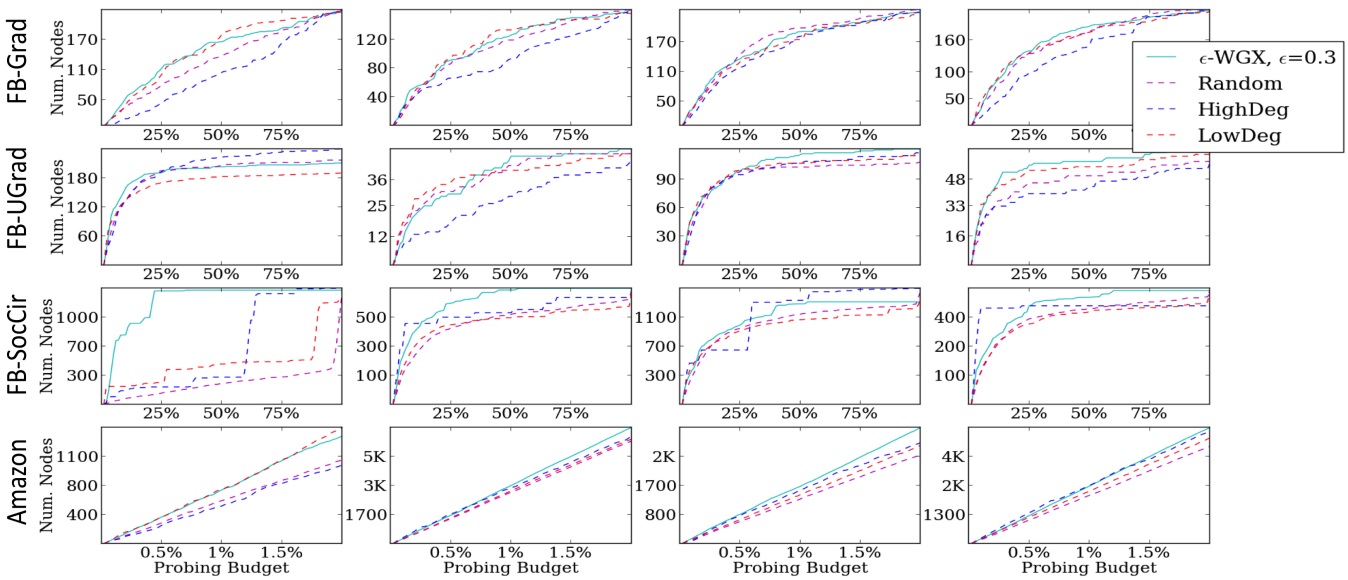

Fig. 7. Results of various strategies on a BFS sample from the FB-SocCir network for the NumNodes reward function. ϵ-WGX outperforms the best baseline strategy. See discussion in Section VI-B for significance testing. (Best viewed in color.)

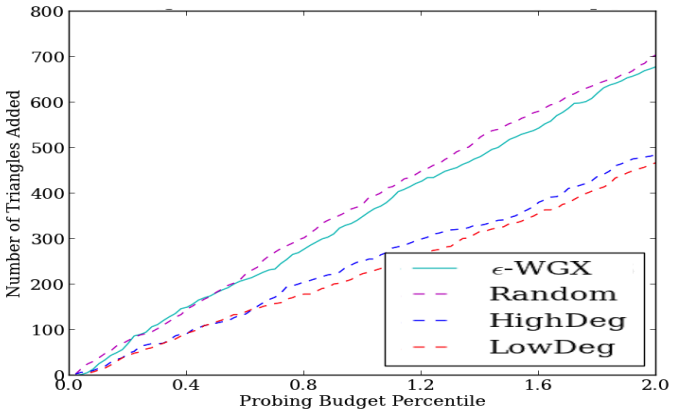

Fig. 10. Results of various strategies on a Random Edge sample from the Amazon network for the TriClose reward function. ϵ-WGX performs comparably with the best baseline strategy. (Best viewed in color.)

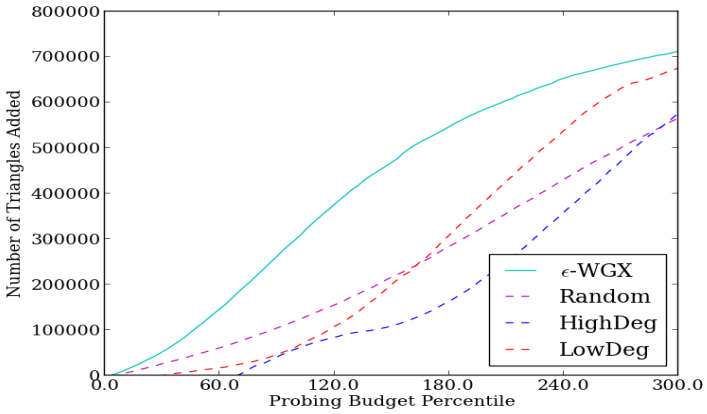

Fig. 11. Results of various strategies on a BFS sample from the FB-SocCir network for the TriClose reward function. ϵ-WGX outperforms the best baseline strategy by a large margin. (Best viewed in color.)

WGX is a safe choice of algorithm. As argued in Section VI-A, it is hard to know which baseline strategy might be the best for a given scenario. Although High Degree happened to be better on average in the exceptions listed above, there are many times when it performs poorly. For example, see Figure 10, which depicts results on the Amazon network with a RandEdge sample, where ϵ-WGX does much better.

These results support our main point, that ϵ-WGX is a safe algorithm, ideal for cases when one does not have background knowledge about which strategy is best for a given network or task.

C. Results for AEP-R Problem

For the most part, the takeaway of results for the AEP-R problem are similar to those on AEP-NR. To save space, we do not present a full array of plots.

For example, see Figure 11, which depicts results on a BFS sample from the FB-SocCir network, for the TriClose reward function. ϵ-WGX algorithm is clearly the best.

As before, Figures 12 and 13 present the mean results, over all samples produced by a given method, at the median budget. These results are presented as a fraction improvement over Random probing. Again, ϵ-WGX is often the best method, and is never worse than second-best.

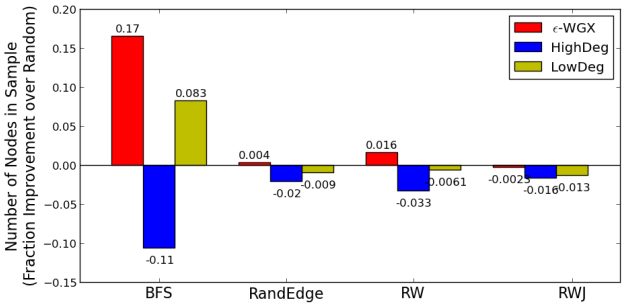

Fig. 12. Mean performance of ϵ-WGX vs baseline strategies, presented as fraction improvement over random, measuring number of nodes in the sample. For all sampling methods except RWJ, ϵ-WGX is the best strategy. For RWJ, it is second-best.

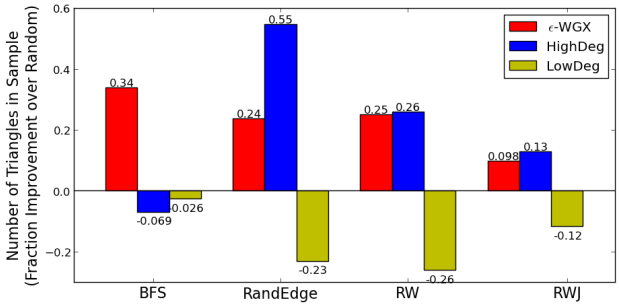

Fig. 13. Mean performance of ϵ-WGX vs baseline strategies, presented as fraction improvement over random, measuring number of triangles in the sample. For BFS samples, ϵ-WGX is the best strategies. For all others, it is second-best.

Performing the Wilcoxon signed-rank test across all networks and probing budgets, we see that results are very similar to those presented in Figures 12 and 13. All differences are statistically significant at $p = 0.01$ except when noted otherwise. ϵ-WGX is the best strategy in all cases, with the following exceptions:

- For the NumNodes reward function, on RWJ samples, Random probing is slightly better than ϵ-WGX. However, this difference is not statistically significant at $p = 0.01$.
- For the TriClose reward function, High Degree probing is the best strategy on RandEdge and RW samples. ϵ-WGX is second best. (Figure 13 shows that High Degree probing is also the best on RWJ samples, but this figure is for only one budget, while the significance tests were run across all budgets.)

For the NumNodes reward function, why does Random probing work so well on the RandEdge and RWJ samples? This occurs because of the nature of the AEP-R problem. In contrast to the other three strategies, the Random probing strategy switches its selection in each step. It is thus more likely to get a new edge each time: although its choices are not necessarily smart, they are new.

D. Effect of Sample Type on Performance

The performance of the probing strategies varies by sample type. In particular, on RandEdge and RWJ samples, Random probing seems to work almost as well as anything else.

A RandEdge sample contains edges sampled uniformly at random. Similarly, a RWJ sample contains many small components. In both cases, most nodes have a high fraction of neighbors outside the sample, so one need not be so strategic about probes. In contrast, a BFS sample is highly skewed: some nodes have many neighbors inside the sample, so node selection is much more important. If one does not know how the sample was generated, ϵ-WGX is a good choice.

E. Running Time Analysis and Efficient Implementation

ϵ-WGX is slower than the baseline probing strategies, but similar to Random probing for larger datasets.[7]

On the small FB-UGrad dataset, we conducted 10,000 probes using the four strategies. On a BFS sample, the High and Low Degree strategies took 0.03 seconds, Random probing took 0.5 seconds, and ϵ-WGX took 1.4 seconds.

However, on a BFS sample from the much larger Amazon network, High Degree probing took 1.4 seconds, Low Degree probing took 0.2 seconds, Random probing took 31.0 seconds, and ϵ-WGX took 34.3 seconds.

The runtime of ϵ-WGX is dominated by the step of selecting a node at random during an exploration step (updating data structures after a probe are constant time operations).

We expect that in practical applications, the cost of conducting the probes (e.g., running a traceroute) in each step will far exceed the cost of selecting the probe. Moreover, assuming that conducting probes has an actual cost, the increase in performance from ϵ-WGX is likely worth the cost of running a random number generator.

VII. Discussion

The results in the previous section have illustrated our two main points: First, predicting the best baseline strategy ahead of time is difficult. Second, given that one does not know ahead of time which strategy will be best for a specific network, sample, reward function, and probing budget, ϵ-WGX is a safe strategy that often performs better than or nearly as well as the best baseline strategy. Although there are cases when a particular baseline strategy is better on average than ϵ-WGX, even in such cases, ϵ-WGX is typically not far behind; moreover, it is successful even in cases when that baseline fails to work well (e.g., Figure 10).

VIII. Related Work

The AEP problem is most similar to topics in graph sampling and crawling; additionally, our ϵ-WGX algorithm draws from the literature on multi-armed bandits.

Graph Crawling and Sampling Much effort has been expended in the area of effectively crawling or sampling

[7] All running time experiments were conducted on a MacBook Pro with a 2.3 GHz Intel Core i7 processor and 8GB of RAM.

large networks. Leskovec and Faloutsos provide an excellent overview of several popular sampling methods [6]. Others have focused on sampling for specific tasks. For example Maiya and Berger-Wolf, as well as Wu, et al., have studied sampling graphs for community detection [8], [9]. Others have used crawling to estimate characteristics of the larger graphs. For example, Maiya and Berger-Wolf use crawling to estimate centrality measures such as PageRank [10], and Cho, et al. propose a method for determining which URLs to examine in a web-crawl so as to maximize the number of nodes visited [11].

Unlike these works, we assume that we are given a graph sample and must then improve it, as opposed to having full control or even knowledge over how the sample was created.

Network Analysis with Limited Information Another question is how to infer characteristics of a graph given a sample. Hanneke and Xing attempt to predict topology given access to only a few nodes [12], and Kim and Leskovec attempt to infer missing pieces of a network [13].

Others have studied problems in which one has a limited budget to gather network information. For example, Avrachenkov, et al. propose a method for locating high-degree nodes in a network using a limited number of queries [14], and O'Brien et al. use local information to estimate the core number of a node [15]. Cohen, et al. show how one can efficiently immunize a networked population in which the network structure is unobserved by targeting neighbors of randomly selected nodes [16]. In the area of sensor networks, Shakkotai considers nodes attempting to find the source of information using only local knowledge [17]. Most similar to our work is that of Macskassy and Provost, showing how one can identify malicious actors in a network by gathering information about those thought to be the most suspicious [18].

Unlike these works, we consider arbitrary reward functions, rather than specific tasks.

Multi-Armed Bandits A multi-armed bandit method is at the heart of theϵ-WGX algorithm. The multi-armed bandit problem dates to the mid-20th century, when Robbins studied the sequential design of experiments [19]. Berry and Fristedt give an overview of bandit problems in [20]. The simplest algorithm for solving the problem is the ϵ-Greedy approach [1], which has been shown to work well in practice [21]. Other methods allow the value of ϵ to decrease as time goes on [21]. Contextual bandit algorithms assign feature vectors to each arm of the bandit; exploration is then guided by these features, so arms with features similar to those of arms that have been successful in the past are more likely to be chosen (e.g., [2]).

A great deal of research has been done on algorithms for the multi-armed bandit problem, and it is not possible for us to cover it all here, but [22] is an excellent survey.

IX. Conclusions

We have presented the **Active Edge Probing** problem, in which one must select nodes in an incomplete sample graph for further exploration, without knowledge of or control over how the sample was generated. We presented the ϵ-WGX algorithm, a multi-armed bandit-based method that identifies nodes that, when probed, produce the most useful information about the complete graph as a whole. Compared against three baseline probing strategies, across a variety of network and sampling methods, ϵ-WGX is always the best or second-best probing strategy. Most critically, ϵ-WGX is uniformly successful; in contrast, even though the baseline strategies are occasionally successful for certain sampling methods or networks, they are inconsistent.

In future work, we are considering other probing scenarios, including cases in which one is only allowed to probe certain nodes.

References

[1] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, 1989.

[2] T. Lu, D. Pal, and M. Pal, "Contextual multi-armed bandits," in *AISTATS*, 2010, pp. 485–492.

[3] E. Samuel, "Sequential maximum likelihood estimation of the size of a population," *Annals of Mathematical Statistics*, vol. 39, no. 3, pp. 1057–1068, 1968.

[4] G. Valiant and P. Valiant, "Estimating the unseen: Estimating the unseen: An n/log(n)-sample estimator for entropy and support size, shown optimal via new clts," in *STOC*, 2011, pp. 685–694.

[5] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," in *WWW*, 2000, pp. 309–320.

[6] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *KDD*, 2006, pp. 631–636.

[7] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: Inferring user profiles in online social networks," in *WSDM*, 2010, pp. 251–300.

[8] A. S. Maiya and T. Berger-Wolf, "Sampling community structure," in *WWW*, 2010, pp. 701–710.

[9] J. Wu, X. Li, L. Jiao, X. Wang, and B. Sun, "Minimum spanning trees for community detection," *Physica A*, vol. 392, no. 9, pp. 2265–2277, 2013.

[10] A. S. Maiya and T. Berger-Wolf, "Online sampling of high centrality individuals in social networks," *PAKDD*, pp. 91–98, 2010.

[11] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through url ordering," in *WWW*, 1998, pp. 161–172.

[12] S. Hanneke and E. P. Xing, "Network completing and survey sampling," in *AISTATS*, 2009, pp. 209–215.

[13] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks," in *SDM*, 2011, pp. 47–58.

[14] K. Avrachenkov, N. Litvak, L. O. Prokhorenkova, and E. Sayargulova, "Quick detection of high-degree entities in large directed networks," in *ICDM*, 2014, pp. 54–65.

[15] M. P. O'Brien and B. Sullivan, "Locally estimating core numbers," in *ICDM*, 2014, pp. 460–469.

[16] R. Cohen, S. Havlin, and D. Ben-Avraham, "Efficient immunization strategies for computer networks and populations," *Phys. Rev. Lett.*, vol. 91, no. 24, p. 247901, 2003.

[17] S. Shakkottai, "Asymptotics of query strategies over a sensor network," in *INFOCOM*, 2004, pp. 548–557.

[18] S. A. Macskassy and F. Provost, "Suspicion scoring based on guilt-by-association, collective inference, and focused data access," in *International Conference on Intelligence Analysis*, 2005.

[19] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. of the AMS*, vol. 58, pp. 527–535, 1952.

[20] D. A. Berry and B. Fristedt, *Bandit problems*. Chapman and Hall Ltd., 1985.

[21] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *ECML*, 2005, pp. 437–448.

[22] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.